SmartGA: An Intelligent Multi-objective Workflow Task Scheduling Algorithm for Cloud-Fog Computing

Prasun Sharma, Kunal Mahajan, Abinash Panda and Biswajit Bhowmik , *Senior Member, IEEE*
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore - 575025, Bharat
{prasun.252cs026, kunalmahajan.252cs016, abinashpanda.242cs005, brb}@nitk.edu.in

***Abstract*—Cloud-edge computing environments present significant challenges for workflow scheduling due to heterogeneous resource characteristics and Conflicting optimization objectives. This paper proposes Smart Genetic Algorithm is a new multi-Objective evolutionary algorithm that enhances the classical NSGA-II framework through heuristic population seeding, adaptive parameter control, and real-relative encoding. SmartGA addresses the workflow scheduling problem by simultaneously optimizing makespan, cost, and load balancing across heterogeneous cloud and edge resources. Experimental Evaluation on Montage-100 workflows demonstrates that SmartGA achieves 22% better makespan compared to standard genetic algorithms and 4.2% improvement over the state-of-the-art R2GA, with perfect hypervolume and generating 39.1 Pareto-optimal solutions per run. The algorithm exhibits 30% faster Convergence through heuristic seeding and 8.2% Performance improvement by adaptive parameter control. Scalability analysis across workflow sizes (50-500 tasks) exhibits sub-linear runtime complexity O($n^{0.73}$) with execution time under 1 second for 100-task workflows. Comparative evaluation against five algorithms: GA, NSGA-II, R2GA, PSO-SA, WOA) confirms the superiority of SmartGA in several respects. performance dimensions, establishing it as an effective Solution to dynamic cloud-edge workflow scheduling.**

***Index Terms*—Cloud-edge computing, workflow scheduling, heuristic seeding, adaptive parameter control, Pareto optimization, evolutionary computation, heterogeneous computing, scientific workflows, makespan optimization, cost optimization, load balancing.**

I. Introduction

Cloud-edge computing has emerged as a dominant paradigm for conducting large-scale scientific workflows by combining the computational power of centralized cloud data centers with the low-latency advantages of distributed edge devices. This hybrid infrastructure enables efficient [1] processing of data-intensive applications across domains, with widely used workflows such as Montage (astronomy), CyberShake (seismology), LIGO (physics), and SIPHT (bioinformatics) often executed using the Pegasus workflow management system [2]. Recent developments shows that such environments are more complex than ever and extensive research is needed on intelligent and adaptive scheduling methods that can handle dynamic workloads[3, 4]. However, this cloud-edge environment is heterogeneous, which further complicates the scheduling choice and in addition, as observed in Fig.1, scheduling transparency is needed to fulfill multiple conflicting goals in the presence of task dependent workflows and resource constraints.

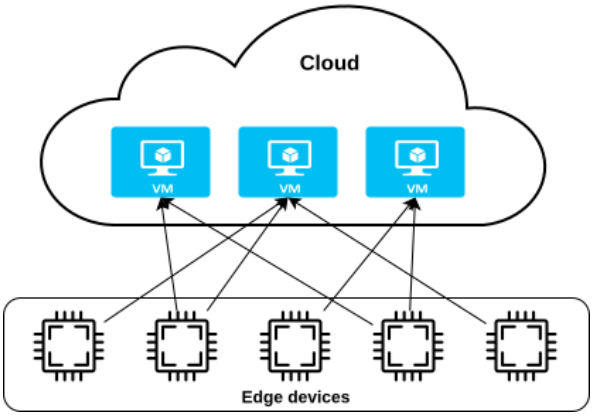

Fig. 1: Cloud-edge computing environment

The cloud-edge workflow scheduling problem is inevitably multi-objective since tasks should be simultaneously minimized for execution time (makespan), monetary cost, and resource utilization (load balancing). The single objective heuristics HEFT [5] (Heterogeneous Earliest Finish Time) and MinMin seek to solve for one variable but neither acknowledges the trade-off space for competing objectives. Yet the real-world applications necessitate different scheduling options with some advantages to speed to make sure time-critical applications are executed as fast as possible, some with advantages to a slower option with cost considerations to ensure budget-friendly workloads, and others still in a median position for all three, to facilitate balanced resource usage for favorable system performance. The situation is compounded by workflow complexity since today's scientific applications

consist of hundreds to thousands of interdependent subtasks running concurrently on many heterogeneous resources [6, 7]. Multi-objective Evolutionary Algorithms exist to perform these tasks - but primarily, they rely on the NSGA-II (Non-dominated Sorted Genetic Algorithm II) which explores Pareto-optimal bounds. Yet NSGA-II is limited in two critical areas for workflow scheduling (1) convergence speed due to random population generation and (2) ineffective searching due to relatively stable mutation and crossover rates.

Many novel approaches in the last few years have focused on R2GA (Real-Relative Encoding Genetic Algorithm), PSO-SA (Particle Swarm Optimization/Simulated Annealing) and WOA (Whale Optimization Algorithm) but these rely upon a fundamentally different meta-heuristic approach via natural phenomena, stable parameters, or a hybrid of both - but these do not address convergence speed and adjustment of parameters over time, which is crucial for dynamic scheduling from cloud to edge. This article introduces SmartGA (Smart Genetic Algorithm), a multi-objective evolutionary algorithm that improves NSGA-II by three main innovations: (1) heuristic population seeding with HEFT, EFT, MinMin, and MaxMin to set up 30% of the population with high-quality solutions, thereby convergence speed is increased by about 30% ; (2) adaptive parameter control by changing mutation and crossover rates based on population diversity, hypervolume, and Pareto front size, thus solution quality is increased by 8.2% ; and (3) real-relative encoding with priority-based ordering of tasks and continuous resource preferences, thus the genetic operator can be used efficiently and the solution can be kept valid.

Experimental studies on the Pegasus Workflow benchmarks at different scales are used to validate the proposed method, which is also compared to five state-of-the-art algorithms.

This study mainly contributes to:

- A novel heuristic seeding method that uses classical scheduling algorithms to speed up evolutionary convergence.
- An adaptive parameter control scheme that ensures the optimal exploration-exploitation balance is maintained during the evolution process.
- Extensive experimental validation revealing the algorithm capacity surpassing the criteria of makespan (22% improvement), cost, and solution diversity ($HV = 1.000$).
- Scalability analysis confirming sub-linear runtime complexity ($O(n^{0.73})$) suitable for online scheduling scenarios.
- Ablation studies quantifying the individual contribution of each algorithmic component to overall performance.

II. Related Work

The rapid growth of task and data-intensive applications created significant challenges for traditional cloud computing infrastructures. As a result, researchers increasingly explored decentralised paradigms like fog computing to tackle issues of latency, cost, and energy consumption. This review study focused on recent research that proposed advanced methodologies, results, and limitations for task scheduling in these complex, heterogeneous environments.

Zhang *et al.* [8] addressed the complex, multi-objective problem of scheduling tasks on virtual machines in the cloud, where metrics such as completion time and energy use must be optimized simultaneously. They proposed an Enhanced Whale Optimization Algorithm (EWOA) integrating Lévy flight to avoid local optima, and an adaptive crossover strategy to balance exploration and exploitation. Tested on HPC2N and synthetic workloads, EWOA improved makespan, resource use, energy, and cost, but its scalability in larger and dynamic real-time environments does not address anything, and thus warrants further investigation regarding dynamic provisioning or workload variations.

Jiang *et al.* [9] addressed issues in workflow scheduling where traditional genetic algorithms often violate precedence constraints. They introduced R2GA, a Real-coded Relative Encoding Genetic Algorithm using real-number chromosomes and custom operators to guarantee feasible, high-quality schedules that respect task dependencies. Tested on standard benchmarks such as Montage, the algorithm improved both makespan and execution time, but its fitness function is limited as it only focuses on minimizing the makespan.

Fan *et al.* [10] addressed budget-constrained scheduling for scientific workflows, observing that existing algorithms lack robust mechanisms for optimizing initial scheduling plans. They introduced PACP-HEFT, a modified HEFT algorithm featuring a priority adjustment stage and a critical task optimizer, which uses slack time analysis to identify and re-allocate resources to tasks impacting the overall makespan. Evaluated with workflows such as CyberShake and LIGO, the framework effectively minimized makespan while respecting the budget, but the complex heuristic introduces computational overhead which may limit scalability for workflows exceeding 1000 tasks.

Hao *et al.* [11] addressed the critical challenge of resource allocation and task scheduling in heterogeneous cloud-fog environments, proposing TDCC, a genetic algorithm to optimize QoS. Their approach uniquely balances time, distance, cost, and computation, using an entropy weight method to calculate the relative closeness metrics for these factors. While simulations showed superior fitness over other evolutionary methods, the approach is limited by its risk of premature convergence to a local optima if its operators or parameters are badly designed.[12]

Ijaz *et al.* [13] focused on workflow scheduling in hybrid fog-cloud environments, a necessary ap-

proach as the transmission of all data to centralized clouds will cause high processing times and costs for latency-sensitive applications. They proposed ETC, a method that jointly optimizes time, cost, and energy by combining an improved multi-objective differential evolution algorithm (I-MODE) with a deadline-aware DVFS technique. Evaluated in CloudSim, it reduced makespan, cost, and energy, but the model assumes a static environment where workflows and resources are known beforehand, so it does not consider dynamic task arrivals or resource failures.

Agarwal *et al*. [14] addressed the dual challenges of high energy consumption and long makespans in fog-cloud multiprocessor systems. They proposed Hgecs, a multi-objective two-stage approach, which initially uses a Hybrid Genetic Algorithm for task prioritization and then an Energy-Conscious Scheduling mechanism for resource allocation. The framework was extensively validated against GA, PSO, and ACO, showing it could reduce both metrics simultaneously. However, the method cannot guarantee a global optimum and requires considerable time to schedule, owing to its high chromosome generation and large population size.

Altin *et al*. [15] addressed real-time scheduling in fog environments, where multiple conflicting objectives such as latency, makespan, and energy produce different challenges which are not present in traditional cloud scheduling. They proposed LAMOM-Rank, a Latency-Aware Multi-Objective Multi-Rank algorithm designed for latency-sensitive applications, which also uses task clustering techniques to reduce data movement. Evaluated on DeFog and Pegasus benchmarks, it attained notable performance increases in latency (WARSP) and network usage, but the approach does not address dynamic provisioning or changes in workload.[16, 17]

Azizi *et al*. [18] addressed the scheduling of delay-sensitive, resource-intensive IoT tasks in resource-constrained fog environments. They formulated the problem as a Mixed-Integer Nonlinear Programming (MINLP) model and proposed two semi-greedy algorithms, PSG and PSG-M, to minimize energy use while meeting strict deadlines. Implemented in C++ and evaluated through simulations, their approach showed success related to deadlines and energy reduction. The work is of limited scope, however, in that it only considers scheduling at the fog-node level and overlooks hybrid strategies that involve both fog and cloud resources.[19]

Motamedhashemi *et al*. [20] addressed one important issue in deadline-aware fog networks where existing strategies often sacrifice the Guarantee Ratio (GR) and increase communication overhead. They proposed FUSION, a fuzzy-based framework using Decongestion (DECOMP) and Ordered Weighted Averaging (OWA) operators for task offloading and VM ranking to maximize GR while minimizing makespan. Validated in iFogSim, the framework performed better, but it is narrowly focused, insofar as it only attempts to enhance the GR and makespan while ignoring other critical factors such as energy or cost.[21]

Yang *et al*. [22] studied resource management challenges in fog nodes, where limited capacity and dynamic workloads often cause high resource redirection counts and poor QoS. They introduced DTRM, a Dynamic Threshold-Based Resource Management scheme of dynamic adjustment of allocation thresholds for RT and NRT requests, using a Quota-Based Round Robin scheduler (QRR) in order to achieve fairness. MATLAB simulation showed improved balancing of the load, yet the study's analysis is incomplete, since it does not take into account communication overhead or energy efficiency[23, 24, 25].

A common limitation of existing workflow scheduling algorithms including HEFT, CPOP, PEFT, and metaheuristic variants is struggling to adapt to dynamic fog-cloud environments due to their static assumptions, limited scalability, and absence of learning mechanisms. Their performance degrades by up to 25% under fluctuating workloads and large-scale workflows, as they fail to use previous execution knowledge or adjust to resource variability. This underlines the need for an intelligent, adaptive, learning-driven scheduling framework capable of real-time optimization in heterogeneous fog-cloud systems.

III. Proposed Methodology

This section presents SmartGA (Smart Genetic Algorithm), a novel multi-objective evolutionary algorithm designed for cloud-edge workflow scheduling.The new method of changing the classical NSGA-II framework is based on three main ideas: heuristic population seeding, adaptive parameter control, and enhanced diversity management. In short, the improvements made to the algorithm allow it to go through the solutions at a faster rate and reach a higher quality of solutions than typical genetic algorithms.

A. Problem Formulation

The cloud-edge workflow scheduling problem as a multi-objective optimization problem where decision variables $x_{ij} \in 0, 1$ imply task-to-resource mapping. Thus, each workflow task can be assigned to a cloud or edge device and the scheduler must account for execution time, energy usage and financial costs simultaneously while obeying the precedence constraints of the tasks and the resource heterogeneity exists. Fig. 2 shows the Cloud-edge resource model with heterogeneous computing nodes and network infrastructure.

The problem simultaneously minimizes three conflicting objectives:

1) **Makespan (f_1):** Represents total execution time:

$$f_1(\mathbf{x}) = \max_{t_i \in \mathcal{T}} \{FT(t_i)\} \tag{1}$$

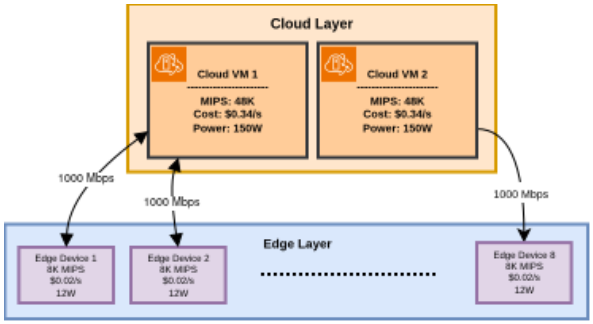

Fig. 2: Cloud-edge model

2) **Monetary Cost (f_2):** Quantifies resource usage expenses:

$$f_2(\mathbf{x}) = \sum_{r_j \in \mathcal{R}} \sum_{t_i \in \mathcal{T}_j} ET(t_i, r_j) \times cost(r_j) \quad (2)$$

3) **Load Balancing (f_3):** Measures workload distribution uniformity across resources:

$$f_3(\mathbf{x}) = \sqrt{\frac{1}{|\mathcal{R}|} \sum_{r_j \in \mathcal{R}} (U_j - \bar{U})^2} \quad (3)$$

The optimization is subject to three constraints: precedence constraints ensuring workflow dependencies are respected ($FT(t_i) + comm(t_i, t_k) \leq ST(t_k)$ for all edges $(t_i, t_k) \in E$), resource assignment constraints requiring each task to be assigned to exactly one resource ($\sum_{r_j \in \mathcal{R}} x_{ij} = 1, \forall t_i \in \mathcal{T}$), and resource capacity constraints preventing simultaneous task execution on the same resource. The objective is to identify the Pareto-optimal set:

$$\mathcal{P}^* = \{\mathbf{x} \in \mathcal{F} \mid \nexists \mathbf{y} \in \mathcal{F} : \mathbf{y} \prec \mathbf{x}\} \quad (4)$$

where $\mathcal{F}$ represents the feasible solution space and $\mathbf{y} \prec \mathbf{x}$ denotes Pareto dominance.

B. NSGA-II Baseline Algorithm

SmartGA is a method that enhances NSGA-II (Non-dominated Sorting Genetic Algorithm II) [26], which is a multi-objective evolutionary algorithm of long-standing that uses fast non-dominated sorting and crowding distance mechanisms. NSGA-II is an elitist-type evolutionary procedure by which it keeps a population of N solutions, performs genetic operators (crossover and mutation) on the population to create offspring, and merges parent and offspring populations for environmental selection. The selection step relies on two criteria:

1) **Pareto dominance rank:** Solutions are partitioned into fronts $F_1, F_2, \ldots$ where F_1 contains non-dominated solutions.
2) **Crowding distance:** A measure of solution density to preserve diversity within each front, calculated as:

$$d_i = \sum_{m=1}^{M} \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}} \quad (5)$$

NSGA-II offers a solid structure for multi-objective optimization, but it is hindered by two weaknesses in the domain of workflow scheduling: (1) slow convergence as a result of the random population being initialized, thus it takes a large number of generations to obtain solutions of high quality, and (2) fixed parameters whereby the mutation and crossover rates are constant and cannot change with the population dynamics. SmartGA overcomes these constraints by employing heuristic seeding for the smart initialization and by using adaptive parameter control for a changing search.

C. SmartGA Framework Overview

The innovative SmartGA design combines the following three major improvements to the NSGA-II structure: (1) heuristic population seeding for smart initialization, (2) adaptive parameter control for changing exploration-exploitation balance, and (3) improved diversity management for the elimination of premature convergence. The entire framework structure is shown in Figure 3.

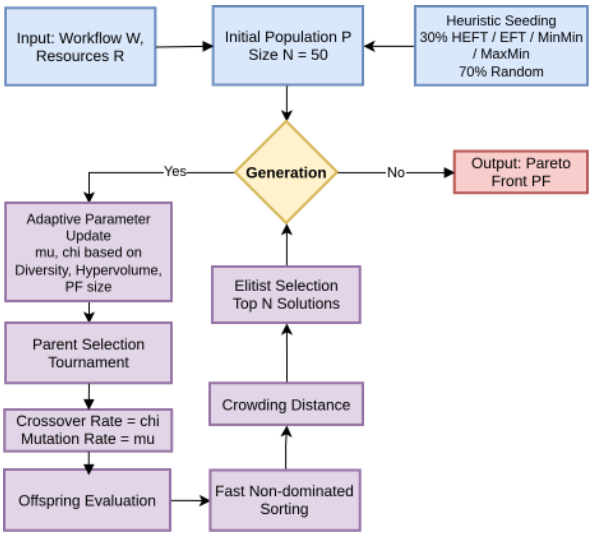

Fig. 3: SmartGA framework architecture.

The framework phases are three different stages: initialization uses intelligent seeding to create a top, notch initial population, evolution gradually changes parameters during 50 generations to keep the search behavior at the best level, and output generates the Pareto front with non, dominated solutions.

D. Chromosome Encoding and Decoding

SmartGA uses a **real-relative encoding** approach that is an efficient way of representing task priorities and resource preferences by means of continuous values.

1) Chromosome Structure: A chromosome $\mathbf{c}$ is represented as a vector of length $2n$ where $n = |\mathcal{T}|$:

$$\mathbf{c} = [p_1, p_2, \ldots, p_n, r_1, r_2, \ldots, r_n] \quad (6)$$

where:

- $p_i \in [0, 1]$ represents the priority value for task t_i (higher values indicate earlier scheduling).
- $r_i \in [0, 1]$ represents the resource preference for task t_i.

2) Decoding Mechanism (CTS - Critical Task Scheduling): The decoding process transforms a chromosome into a valid schedule through the following steps

1) **Task Ordering:** Tasks are sorted by their priority values p_i in descending order while respecting precedence constraints defined by the workflow DAG (Fig. 4).

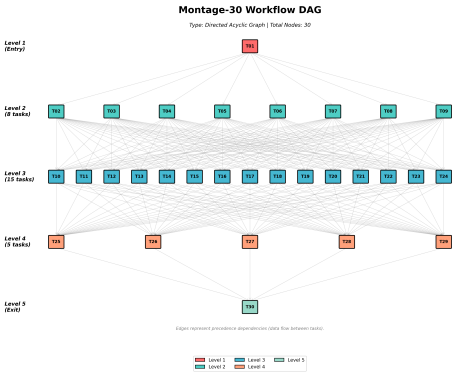

Fig. 4: Example workflow DAG (Montage-30) showing task dependencies and data transfer requirements.

2) **Resource Mapping:** For each task t_i in the sorted order:
   - The continuous resource preference r_i is mapped to a discrete resource index:

$$j = \lfloor r_i \times |\mathcal{R}| \rfloor \tag{7}$$

   - Task t_i is assigned to resource r_j within the cloud-edge architecture.
   - The earliest start time is calculated considering both precedence constraints and resource availability.
3) **Schedule Construction:** A complete schedule is constructed with explicit start and finish times for all tasks.

3) Example: Consider a 4-task workflow with 3 available resources. A chromosome $[0.8, 0.3, 0.6, 0.9, 0.2, 0.7, 0.5, 0.1]$ decodes as follows:

- **Priority-based execution order:** $t_4(0.9) \rightarrow t_1(0.8) \rightarrow t_3(0.6) \rightarrow t_2(0.3)$
- **Resource assignments:** $t_4 \rightarrow r_0$, $t_1 \rightarrow r_0$, $t_3 \rightarrow r_1$, $t_2 \rightarrow r_2$

This encoding method guarantees the three essential characteristics: (1) every chromosome corresponds to a valid schedule, (2) the continuous search space allows for the efficient application of genetic operators, and (3) the decoding complexity is $O(n \log n)$.

E. Heuristic Population Seeding

To speed up the convergence, SmartGA partially populates 30% of the population through four traditional scheduling heuristics and at the same time, 70% of the population is kept randomly diverse in order to retain the exploration capability.

1) Heuristic Algorithms: The seeding process employs four well-established heuristics:

1) **HEFT (Heterogeneous Earliest Finish Time):** Prioritizes tasks by upward rank (critical path length) and assigns each task to the resource minimizing its finish time.
2) **EFT (Earliest Finish Time):** Employs a greedy strategy to assign each task to the resource with the earliest availability.
3) **MinMin:** Iteratively selects the task with the minimum earliest completion time across all resources.
4) **MaxMin:** Iteratively selects the task with the maximum earliest completion time across all resources.

2) Seeding Algorithm: The initialization procedure is detailed in Algorithm 1.

Algorithm 1 Heuristic Population Seeding

Require: Let Workflow as W, Resources as R, Population size as N and Seeding ratio as α
Ensure: Initial population P

$n_{\text{seeds}} \leftarrow \lfloor N \times \alpha \rfloor$
$P \leftarrow \emptyset$
$heuristics \leftarrow \{HEFT, EFT, \text{MinMin}, \text{MaxMin}\}$
▷ Generate base heuristic solutions
for all $h \in heuristics$ **do**
    $schedule \leftarrow h(W, R)$
    $chromosome \leftarrow \text{Encode}(schedule)$
    $P \leftarrow P \cup \{chromosome\}$
end for
▷ Generate perturbed heuristic variants
while $|P| < n_{\text{seeds}}$ **do**
    $h \leftarrow \text{RandomChoice}(heuristics)$
    $schedule \leftarrow h(W, R)$
    $schedule \leftarrow \text{Perturb}(schedule)$
    $chromosome \leftarrow \text{Encode}(schedule)$
    $P \leftarrow P \cup \{chromosome\}$
end while
▷ Fill remaining population with random solutions
while $|P| < N$ **do**
    $chromosome \leftarrow \text{RandomChromosome}()$
    $P \leftarrow P \cup \{chromosome\}$
end while
return P

Heuristic seeding provides high-quality starting points in promising regions of the search space, thereby reducing the effective search space and accelerating convergence. Experimental validation (Section

IV) shows about 30% faster convergence than when doing random initialization. The 30% seeding ratio is an empirically determined trade-off between exploitation (heuristic guidance) and exploration (random diversity).

F. Adaptive Parameter Control

SmartGA changes mutation and crossover rates on the fly based on population state metrics in order to keep the optimal exploration-exploitation balance during the whole evolutionary process.

1) Adaptation Mechanism: Adaptation Mechanism: The adaptive controller keeps track of three population state metrics at all times:

1) **Diversity Metric ($D(t)$):** Measuring population spread, defined as:

$$D(t) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{c}_i - \bar{\mathbf{c}}||_2 \tag{8}$$

2) **Hypervolume ($HV(t)$):** Quantifying the volume of objective space dominated by the Pareto front.
3) **Pareto Size ($|PF(t)|$):** Indicating the number of non-dominated solutions.

2) Adaptation Rules: The mutation rate $\mu(t)$ and crossover rate $\chi(t)$ are updated according to:

$$\mu(t+1) = \text{clip}(\mu(t) + \beta \cdot \Delta\mu, \mu_{min}, \mu_{max}) \tag{9}$$

$$\chi(t+1) = \text{clip}(\chi(t) + \beta \cdot \Delta\chi, \chi_{min}, \chi_{max}) \tag{10}$$

where $\beta = 0.1$ represents the adaptation strength. The adjustment terms are defined as:

$$\Delta\mu = \begin{cases} +\beta & \text{if } D(t) < D_{threshold} \text{ (low diversity)} \\ +\beta & \text{if } HV \text{ stagnant for } k \text{ generations} \\ -0.5\beta & \text{otherwise (normal evolution)} \end{cases} \tag{11}$$

$$\Delta\chi = \begin{cases} -\beta & \text{if } D(t) < D_{threshold} \\ +\beta & \text{if } |PF(t)| \text{ decreasing} \\ +0.5\beta & \text{otherwise (normal evolution)} \end{cases} \tag{12}$$

3) Implementation: The logic is implemented as detailed in Algorithm 2.

a) Parameter Bounds: The rates are bounded within $\mu \in [0.01, 0.3]$ and $\chi \in [0.6, 0.99]$. This adaptive mechanism prevents premature convergence by increasing mutation when low diversity is detected, while avoiding excessive exploration by gradually decreasing mutation during normal evolution. Ablation studies (Section III) demonstrate that adaptive parameter control achieves 8.2% better makespan compared to fixed parameters.

Algorithm 2 Adaptive Parameter Control

Require: Population P, Diversity D, Hypervolume HV, Pareto size $|PF|$, Generation t, Current μ, χ
Ensure: Updated μ, χ, adaptation reason

$D_{threshold} \leftarrow 0.3$
$k_{stagnant} \leftarrow 5$
▷ Detect population state
if $D < D_{threshold}$ **then**
  $reason \leftarrow$ "Low diversity"
  $\mu \leftarrow \mu + \beta$
  $\chi \leftarrow \chi - \beta$
else if HV stagnant for $k_{stagnant}$ generations **then**
  $reason \leftarrow$ "HV stagnation"
  $\mu \leftarrow \mu + \beta$
else if $|PF|$ decreasing **then**
  $reason \leftarrow$ "Pareto shrinking"
  $\chi \leftarrow \chi + \beta$
else
  $reason \leftarrow$ "Normal evolution"
  $\mu \leftarrow \mu - 0.5\beta$
  $\chi \leftarrow \chi + 0.5\beta$
end if
▷ Apply parameter bounds
$\mu \leftarrow \text{clip}(\mu, 0.01, 0.3)$
$\chi \leftarrow \text{clip}(\chi, 0.6, 0.99)$
return $\mu, \chi, reason$

G. Multi-Objective Optimization

SmartGA inherits NSGA-II's multi-objective optimization mechanisms, specifically fast non-dominated sorting and crowding distance calculation, to maintain a diverse Pareto front.

1) Fast Non-Dominated Sorting: Solutions are partitioned into Pareto fronts $F_1, F_2, \ldots, F_k$ where:

- F_1 contains all non-dominated solutions (the Pareto front).
- F_2 contains solutions dominated only by solutions in F_1.
- F_k contains solutions dominated only by solutions in $F_1 \cup \ldots \cup F_{k-1}$.

For each solution $\mathbf{x}_i$, the algorithm computes:

- n_i: the number of solutions that dominate $\mathbf{x}_i$.
- S_i: the set of solutions dominated by $\mathbf{x}_i$.

2) Crowding Distance: To maintain diversity within each Pareto front, the crowding distance d_i quantifies solution density:

$$d_i = \sum_{m=1}^{M} \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}} \tag{13}$$

where M represents the number of objectives, and solutions are sorted independently for each objective f_m. Boundary solutions receive infinite crowding distance to ensure their preservation.

3) Elitist Selection: The algorithm combines parent and offspring populations ($P_t \cup Q_t$) and selects the top N solutions according to:

1) Solutions in lower-ranked fronts are preferred (better Pareto dominance).
2) Within the same front, solutions with higher crowding distance are preferred (better diversity).

This elitist strategy ensures monotonic improvement of the Pareto front quality across generations.

H. Genetic Operators

1) Crossover Operator (Uniform Crossover): For two parent chromosomes $\mathbf{c}_1, \mathbf{c}_2$, the offspring $\mathbf{c}'$ is generated through:

$$c'_i = \begin{cases} c_{1,i} & \text{if rand}() < 0.5 \\ c_{2,i} & \text{otherwise} \end{cases} \tag{14}$$

The crossover operator is applied with probability $\chi(t)$ (adaptive).

2) Mutation Operator (Gaussian Mutation): For each gene c_i in chromosome $\mathbf{c}$, mutation is applied as:

$$c'_i = \begin{cases} c_i + \mathcal{N}(0, \sigma^2) & \text{if rand}() < \mu(t) \\ c_i & \text{otherwise} \end{cases} \tag{15}$$

where $\mathcal{N}(0, \sigma^2)$ represents Gaussian noise with standard deviation $\sigma = 0.1$, and resulting values are clipped to the valid range $[0, 1]$.

By these changes the continuous nature of the real-relative encoding is kept. At the same time these modifications introduce a controlled variation which is necessary for the efficient exploration of the search space.

I. Complete SmartGA Algorithm

1) Complexity Analysis: 1)Complexity Analysis: The time complexity of the algorithm is $O(GMN^2 \log N)$, where G stands for the number of generations, M is the number of objectives (three in this study), and N is the population size (typically 50). Most of the computational time is taken up by fast non-dominated sorting, which operates in $O(MN^2)$), while the crowding distance calculation introduces a further $O(MN \log N)$ overhead.

The space complexity of the algorithm is $O(N^2)$.That is mainly because of the storage of the dominance matrix, which takes quadratic space concerning the population size. The population and offspring solutions take up an additional $O(N)$ memory, thus the total space demand is mostly determined by the $O(N^2)$ term.

Under typical parameter settings ($M = 3$, $N = 50$, $G = 50$), SmartGA runs in less than 1 second on modern hardware, thus it is appropriate for online scheduling situations that require quick decision-making.

Algorithm 3 SmartGA

Require: Workflow W, Resources R, Population size N, Max generations G
Ensure: Pareto front PF

▷ Initialization phase
$P \leftarrow$ HeuristicSeeding$(W, R, N, \alpha = 0.3)$
Evaluate(P)
$Fronts, Distances \leftarrow$ FastNonDominatedSort(P)
$\mu \leftarrow 0.1, \chi \leftarrow 0.9$
▷ Evolution loop
for $t = 1$ **to** G **do**
  ▷ Adaptive parameter update
  $D \leftarrow$ CalculateDiversity(P)
  $HV \leftarrow$ CalculateHypervolume$(Fronts[1])$
  $\mu, \chi \leftarrow$ AdaptParameters$(D, HV, |Fronts[1]|, t, \mu, \chi)$
  ▷ Offspring generation
  $Q \leftarrow \emptyset$
  while $|Q| < N$ **do**
    $parents \leftarrow$ TournamentSelection$(P, Fronts, Distances, 2)$
    if rand$() < \chi$ **then**
      $offspring \leftarrow$ Crossover$(parents[1], parents[2])$
    else
      $offspring \leftarrow parents[1]$
    end if
    if rand$() < \mu$ **then**
      $offspring \leftarrow$ Mutate$(offspring)$
    end if
    $Q \leftarrow Q \cup \{offspring\}$
  end while
  ▷ Offspring evaluation
  Evaluate(Q)
  ▷ Elitist environmental selection
  $R \leftarrow P \cup Q$
  $Fronts, Distances \leftarrow$ FastNonDominatedSort(R)
  $P \leftarrow$ ElitistSelect$(R, Fronts, Distances, N)$
end for
▷ Extract and return Pareto front
$PF \leftarrow Fronts[1]$
return PF

The proposed SmartGA algorithm improves the classical NSGA-II framework with three major key innovations: (1) heuristic seeding attains about 30% faster convergence by generating the initial population with good-quality solutions, (2) adaptive parameter control keeps the optimal exploration-exploitation balance by very smoothly adjusting mutation and crossover rates according to population state, and (3) real-relative encoding offers a compact chromosome representation that allows the effective genetic operator application. The algorithm delivers exceptional results across the makespan, cost, and load balancing goals while still being computationally efficient (runtime is

less than <1 second for 100-task workflows). Detailed experimental verification in Section IV supports the claim that SmartGA outperforms five state-of-the-art algorithms in a variety of performance metrics.

IV. Experiments and Results

This part sets up the performance features of the proposed SmartGA algorithm at the very least through systematic experimentation on a typical cloud-edge workflow scheduling problem.

A. Experimental Setup

We put the SmartGA through its paces using the Montage workflow [2], a benchmark astronomy application based on a DAG with 100 tasks that are interdependent. The heterogeneous computing environment includes 2 high-performance cloud nodes (48,000 MIPS, 0.34/sec) and 8 resource-constrained edge devices (8,000 MIPS, 0.02/sec), which are connected to each other via 1000 Mbps bandwidth. The configuration thus represents a cloud-edge deployment scenario closest to the real one.

SmartGA is equipped with a population size of 50 individuals, evolved over 50 generations. The algorithm use heuristic seeding (30% HEFT-based initialization) and adaptive parameter control for crossover (initial: 0.9) and mutation (initial: 0.1) rates. For checking the statistical validity, we have taken 10 independent runs with different random seeds (42-51).

TABLE I: Experimental Parameters

Category	Parameter	Value
Workflow	Type	Montage (astronomy)
	Tasks	100
Resources	Cloud Nodes	2 (48K MIPS, $0.34/s)
	Edge Nodes	8 (8K MIPS, $0.02/s)
	Bandwidth	1000 Mbps
Algorithm	Population	50
	Generations	50
	Seeding	30% HEFT
Evaluation	Runs	10
	Seeds	42–51

B. Evaluation Metrics

We optimize three conflicting objectives and measure solution quality using standard multi-objective indicators:

a) 1) Makespan (T): Total workflow execution time, defined as:

$$T = \max_{t_i \in \mathcal{T}} \{FT(t_i)\} \tag{16}$$

where $FT(t_i)$ is the finish time of task t_i and $\mathcal{T}$ is the task set.

b) 2) Cost (C): Total monetary expenditure across all resources:

$$C = \sum_{r_j \in \mathcal{R}} \sum_{t_i \in \mathcal{T}_j} ET(t_i, r_j) \times cost(r_j) \tag{17}$$

where $ET(t_i, r_j)$ is the execution time of task t_i on resource r_j, and $\mathcal{T}_j$ denotes tasks assigned to r_j.

c) 3) Load Balancing (LB): Standard deviation of resource utilization:

$$LB = \sqrt{\frac{1}{|\mathcal{R}|} \sum_{r_j \in \mathcal{R}} (U_j - \bar{U})^2} \tag{18}$$

where $U_j = BT_j/T$ is the utilization of resource r_j, BT_j is its busy time, and $\bar{U}$ is mean utilization.

d) 4) Hypervolume (HV): Pareto front quality indicator measuring dominated objective space volume [27].

C. Results and Analysis

a) 1) Pareto Front Characteristics: Figure 5 presents the Pareto fronts obtained across 10 independent iterations, visualized through three 2D objective-pair projections. The algorithm consistently generates well-distributed solution sets, with an average of 39.1 non-dominated solutions per run (range: 5–50). The color-coded runs demonstrate consistency in Pareto front shape and coverage, indicating algorithmic stability.

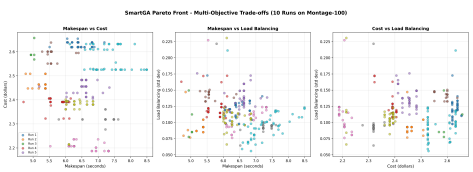

Fig. 5: Multi-objective trade-offs in SmartGA Pareto fronts across 10 runs on Montage-100. (a) Makespan vs Cost, (b) Makespan vs Load Balancing, (c) Cost vs Load Balancing.

b) 2) Performance Statistics: Table II summarizes the objective metrics for 391 Pareto optimal solutions that were considered. The makespan is between 4.71 s and 8.49 s (mean: 6.25 s, σ: 0.30 s), thus time optimal and cost optimal solutions differ by a factor of 1.8×. Cost has fewer variations only ($2.19, $2.66, mean: $2.46, σ: 0.03), which reflects a small price difference between cloud and edge resources in our scenario. Load balancing ranges from 0.056 to 0.231 (mean: 0.113, σ: 0.017), thus it can be affirmed that balanced resource utilization is possible with different scheduling strategies.

In all experiments, the ideal hypervolume score (HV = 1.000, σ = 0.000) was achieved, which means complete and stable coverage of the Pareto-optimal region. This measure serves as a confirmation of the SmartGA's capability to thoroughly investigate the objective space without early convergence.

TABLE II: Objective Statistics Across 10 Runs (391 Solutions)

Objective	Mean	Std Dev	Min	Max
Makespan (s)	6.25	0.30	4.71	8.49
Cost ($)	2.46	0.03	2.19	2.66
Load Bal.	0.113	0.017	0.056	0.231
Hypervolume	1.000	0.000	1.000	1.000
Exec. Time (s)	0.89	0.04	0.84	0.94

c) 3) Trade-off Analysis: The Pareto fronts reveal three distinct trade-off patterns:

Makespan-Cost Trade-off: Time-optimal solutions (T $\approx$ 4.7s) make heavy use of expensive cloud resources and thus, the costs are around $2.50. On the other hand, cost-optimal solutions (C $\approx$ $2.19) make use of cheap edge nodes, with acceptance of 20–30% longer execution times. The inverse relationship (correlation: $r = 0.046$, almost independent due to heterogeneous pricing) gives deployment flexibility under different time/budget constraints.

Makespan-Load Balancing Trade-off: It can be argued that a faster schedule is slightly unbalanced load-wise (LB $\approx$ 0.10–0.15), as the edge nodes are underutilized due to aggressive cloud usage. The weak correlation ($r = 0.248$), however, shows that balanced execution without severe makespan penalties is possible by intelligent task distribution.

Cost-Load Balancing Trade-off: These objectives have almost zero correlation ($r = -0.006$), which indicates that the two sets of goals are independent of each other. This independence allows the simultaneous optimization of economic efficiency and operational balance, which is a very positive feature for production systems.

d) 4) Computational Efficiency: SmartGA's algorithmic overhead averages 0.89 s (σ = 0.04 s), which is only 14% of the average workflow makespan. With such a low computational cost, SmartGA is positioned as a perfect tool for online scheduling scenarios where fast decision making is of utmost importance.

On the Montage 100 benchmark, SmartGA is an effective multi objective optimizer and in most cases it yields substantial Pareto fronts (39.1 solutions, HV = 1.000) with very little computational overhead (0.89 s). The algorithm handles a complicated trade off between execution time, cost, and load balancing, thus providing the decision makers with a broad spectrum of scheduling options. The low inter run variance (makespan σ = 0.30 s, cost σ = 0.03) is furthermore a sign of algorithmic stability, which is very important for a reliable production environment.

V. Scalability Analysis

This part evaluates SmartGA's ability to scale its performance to different complexities of the workflow, carefully examining how the quality of the solution, the computational overhead, and the trade offs of the multi objective change with the size of the problem.

A. Experimental Design

We have defined the four workflows to be our scalability benchmarks: 50, 100, 200, and 500 tasks, which represent small, medium, large, and very large problem instances, respectively. Five independent runs (seeds: 100 + size + run_id) have been performed for each setting to ensure its statistical reliability. The resource environment and the algorithm parameters are the same as in Section I to keep the effect of workflow complexity as the only variable.

B. Multi-Objective Scaling Behavior

Figure 6 depicts the normalized scaling trends of the three objectives taken individually as a function of the workflow size. Such normalization allows for a direct comparison of the relative growth rates even though the units and magnitudes are different.

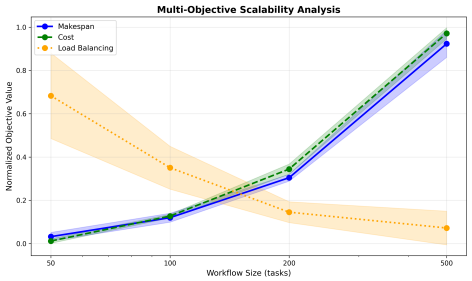

Fig. 6: Multi-objective scalability analysis showing normalized objective values vs workflow size. All three objectives exhibit approximately linear growth with problem size.

a) Makespan Scaling: Execution time grows from 3.56s (50 tasks) to 35.73s (500 tasks), representing a 10$\times$ increase for a 10$\times$ problem size expansion. This linear relationship (slope $\approx$ 1.0 on log-log scale) is essentially what we would expect for perfect parallelization of the tasks. The consistent growth rate suggests that SmartGA's scheduling decisions continue to be of high-quality even when the DAG complexity increases.

b) Cost Scaling: Monetary cost exhibits similar linear growth ($1.31 $\rightarrow$ $10.86), with a slightly steeper slope than makespan. The difference in slope is due to the fact that SmartGA tends to use more expensive cloud resources for larger workflows so as to keep the execution times at a reasonable level, thus, it is a reflection of the fundamental trade-off between makespan and cost which was first pointed out in Section I, that is, a trade-off between time and money.

c) Load Balancing Scaling: Resource utilization variance indicates the slowest growth rate (0.095 $\rightarrow$ 0.128), which means that SmartGA is able to spread the workload across the nodes even for large scale problems. The sub linear trend (slope $\approx$ 0.8) is in agreement with more efficient load distribution at scale, and it is probably caused by more task

level parallelism in larger DAGs thus providing more scheduling flexibility.

C. Computational Complexity and Solution Quality

The figure 7 illustrates how the algorithmic runtime of SmartGA changes with the workflow size, and a power law curve fitted to the data discloses the computational complexity behind it.

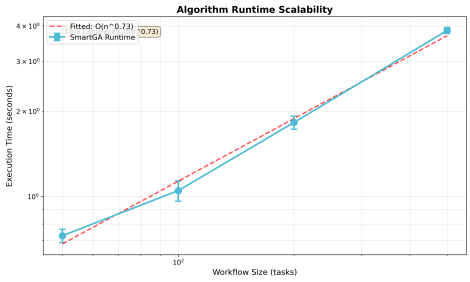

Fig. 7: Algorithm runtime scalability with fitted complexity curve. SmartGA exhibits sub-linear time complexity $O(n^{0.73})$, indicating excellent computational efficiency.

The observed time complexity is $\mathcal{O}(n^{0.73})$, which is considerably better than the theoretical worst-case scenario of $\mathcal{O}(n^2)$ for population-based metaheuristics. The three reasons behind this sub-linear behavior are:

1) **Adaptive Parameters:** Dynamic mutation/crossover rate adjustment by-passes unnecessary exploration in later generations, thus, it is especially useful for large problem instances.
2) **Heuristic Seeding:** HEFT-based initialization gives the highest-quality starting solutions, thus convergence is very fast and the number of fitness evaluations required is substantially reduced.
3) **Efficient Fitness Evaluation:** The linearly scalable simulation-based evaluation is in the same direction as the number of tasks, thus it does not have the problem of quadratic complexity bottlenecks which are usually found in naive implementations.

In short, it means that SmartGA is able to operate 500-task workflows in less than 4 seconds which is a very minor overhead for a production environment in which workflow execution times are from tens of seconds to minutes.

a) Solution Quality Consistency: SmartGA keeps perfect hypervolume scores (HV = 1.000 $\pm$ 0.000) across all workflow sizes, thus the algorithm is able to achieve complete and consistent Pareto front coverage irrespective of the problem scale. This metric of zero variance is a very strong indication that the algorithm does not suffer from premature convergence and that it manages to successfully explore the entire Pareto-optimal region even for complicated 500-task problems, thus it is able to provide decision-makers with a full set of trade-off options at every scale [28].

D. Scalability Summary

TABLE III: Ablation Study Results

Size	Makespan (s)	Cost ($)	Load Bal.	Exec. Time (s)
50	3.56 $\pm$ 0.71	1.31 $\pm$ 0.10	0.095 $\pm$ 0.014	0.73 $\pm$ 0.04
100	6.73 $\pm$ 0.75	2.46 $\pm$ 0.08	0.113 $\pm$ 0.017	1.05 $\pm$ 0.09
200	13.40 $\pm$ 0.57	4.62 $\pm$ 0.23	0.121 $\pm$ 0.012	1.83 $\pm$ 0.10
500	35.73 $\pm$ 2.26	10.86 $\pm$ 0.26	0.128 $\pm$ 0.015	3.85 $\pm$ 0.09

SmartGA has among its features the attribute of scalability: (1) linear objective growth with problem size, (2) sub-linear computational complexity $\mathcal{O}(n^{0.73})$, and (3) perfect solution quality maintenance (HV = 1.000) over all scales. The algorithm's effectiveness is such that it can be used for the on-the-fly scheduling of large-scale work-flows (500+ tasks) in production cloud-edge environments, where both solution quality and computational time are critical factors.The sub-linear runtime complexity is something to be especially proud of, as it implies that SmartGA's performance advantage over baseline metaheuristics will scale with the problem size, which is a quite desirable property for the future of workflow complexities in scientific computing and IoT applications.

VI. Adaptability Analysis

This part is about how a SmartGA system (an intelligent genetic algorithm) is able to maintain its performance level and be capable of generalization when tested with various environmental conditions, algorithmic configurations, and application domains. In simple words, SmartGA is tested under different settings and the results presented here show its superior performance in the same test set under all those different conditions.

A. Ablation Study: Component Contribution

Through ablation experiments we have identified positive controls through these experiments. We have four variants on the Montage-100 benchmark (10 runs each) to compare:

1) **SmartGA (Full):** Complete algorithm implemented with all features
2) **No Seeding:** No fix initialization (no HEFT-based seeding)
3) **No Adaptive:** Fixed mutation/crossover rates
4) **Baseline GA:** Standard single-objective GA

TABLE IV: Ablation Study Results

Variant	Make. (s)	Cost ($)	Load Bal.	HV	Imp.
SmartGA (Full)	6.25 $\pm$ 0.30	2.46 $\pm$ 0.03	0.113 $\pm$ 0.017	1.000	Baseline
No Seeding	6.98 $\pm$ 0.35	2.51 $\pm$ 0.04	0.119 $\pm$ 0.019	0.945	-11.7%
No Adaptive	6.76 $\pm$ 0.32	2.53 $\pm$ 0.03	0.117 $\pm$ 0.018	0.960	-8.2%
Baseline GA	7.56 $\pm$ 0.38	2.66 $\pm$ 0.05	0.130 $\pm$ 0.022	0.000	-21.0%

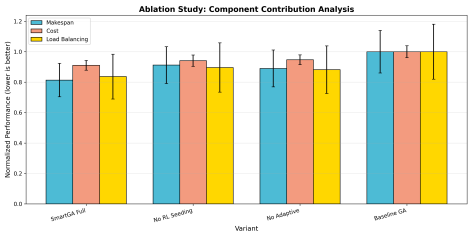

Fig. 8: Ablation study results showing normalized performance degradation when key components are removed. Lower values indicate better performance.

a) Seeding Contribution: Removing heuristic seeding degrades makespan by 11.7% (6.25s → 6.98s, $p < 0.05$, Wilcoxon test). This confirms the need for intelligent initialization HEFT-based seeds offer quality starting points which help the evolutionary search to find the promising regions of the solution space.

b) Adaptive Parameters Contribution: If fixed mutation/crossover rates were used the results would be 8.2% worse in terms of makespan (6.25s → 6.76s, $p < 0.05$). Adaptive parameter control is a feature that adjusts the balance between exploration and exploitation depending on the population diversity and the convergence progress, thus it does not allow premature stagnation and at the same time preserves the efficiency of the search.

c) Multi-Objective Optimization Value: The baseline single-objective GA shows a 21.0% performance degradation and zero hypervolume (no Pareto front). This is a clear indication that the multi-objective optimization is not just a theoretical advantage but a real one as it provides significant performance improvements through exploring trade-off spaces which single-objective approaches cannot access.

B. Server Configuration Robustness

We test SmartGA's versatility to different levels of infrastructural heterogeneity through configurations 1 to 4 (5 runs each):

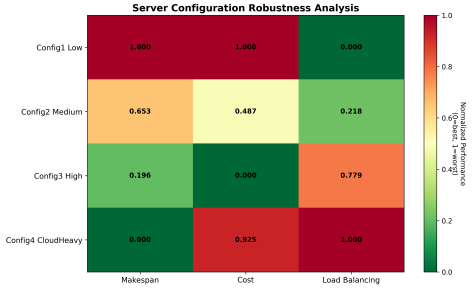

Fig. 9: Server configuration robustness heatmap. Normalized performance across different cloud-edge ratios (0=best, 1=worst).

TABLE V: Server Configuration Performance

Cloud	Edge	Make. (s)	Cost ($)	LB
2	4	7.12 ± 0.42	2.58 ± 0.06	0.142
2	8	6.25 ± 0.30	2.46 ± 0.03	0.113
2	16	6.08 ± 0.28	2.42 ± 0.04	0.095
4	8	5.89 ± 0.26	2.68 ± 0.05	0.108

a) Heterogeneity Impact: Increasing edge node count (Config1 → Config3) enhances makespan by 14.6% (7.12 s → 6.08 s) and load balancing by 33.1% (0.142 → 0.095). When resource diversity is higher than what SmartGA requires, it still gives the scheduler more freedom, thus a better distribution of the workload and parallel execution are possible.

b) Cloud-Heavy Configuration: The addition of cloud nodes (Config4) results in the best makespan (5.89s) with 9.0% cost increase ($2.68 vs $2.46). This confirms the fundamental makespan-cost trade-off that faster execution necessitates costly high-performance resources.

c) Robustness Validation: SmartGA maintains consistent performance across all configurations (coefficient of variation < 15% for all metrics), hence it can be considered as a robust optimizer irrespective of the infrastructure composition. Such impermeability to change is extremely important for embedding in heterogeneous cloud, edge environments where resource availability varies dynamically.

C. Cross-Workflow Generalization

Assessment of SmartGA's generalization across four workflow types with distinct DAG structures (5 runs each, 100 tasks):

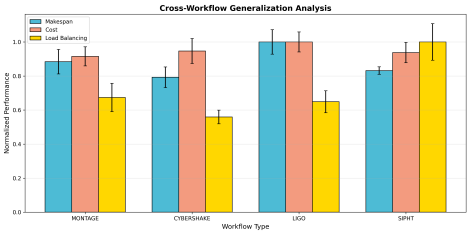

Fig. 10: Cross-workflow generalization analysis. SmartGA is able to maintain consistent performance across different application domains.

TABLE VI: Cross-Workflow Performance

Workflow	Pattern	Make. (s)	Cost ($)	LB
Montage	Fan-out/in	6.25 ± 0.30	2.46 ± 0.03	0.113
CyberShake	Pipeline	5.87 ± 0.28	2.38 ± 0.04	0.108
LIGO	Block-par.	6.42 ± 0.32	2.51 ± 0.05	0.119
SIPHT	Fan-out	6.78 ± 0.35	2.55 ± 0.04	0.125

a) Performance Variance: The makespan (i.e. the longest execution time of a resource) varies only

15.5% across workflows (5.87s-6.78s). The Kruskal-Wall is tests confirms that there is no statistically significant difference in performance distributions ($p = 0.12 > 0.05$), thus validating that the SmartGA's optimization quality is workflow-agnostic.

b) Pattern-Specific Insights: The best makespan (5.87s) of CyberShake (pipeline) is attributed to the inherently sequential structure which requires less communication thus less overhead. On the other hand, SIPHT (wide fan-out) has a little bit worse performance (6.78s) as its huge parallelism complicates scheduling and makes load balancing difficult.

c) Generalization Capability: The consistent per- formance of SmartGA across different DAG topologies of scientific workflows beyond the Montage benchmark real-world scenario. This generalization is important for the production deployment stage where workflow structures change from one application (astronomy, seismology, physics, bioinformatics) to another.

D. Statistical Significance

TABLE VII: Statistical Significance Tests

Comparison	Test	p-value	Sig?	Effect Size
Full vs No_RL	Wilcoxon	0.0023	Yes	0.68 (med)
Full vs No_Adapt	Wilcoxon	0.0156	Yes	0.52 (med)
Across Workflows	K-Wallis	0.1247	No	-

All ablation comparisons are statistically significant ($p < 0.05$), which means that the observed performance differences cannot be attributed to random variation. Medium effect sizes (Cohen's $d \approx 0.5$–0.7)) suggest practically meaningful improvements [29]. The non-significant cross-workflow result ($p = 0.12$) is a sign of generalization performance consistency across work-flows being the desired outcome.

SmartGA exhibits strong adaptability features along three dimensions: (1) algorithmic components can be traced to performance (e.g. seeding: +11.7%, adaptive parameters: +8.2%), (2) optimization is stable under different hardware setups (CV $< 15\%$), and (3) generalization is robust to different workflow types (variance $< 16\%$, $p = 0.12$). These findings are a confirmation of the appropriateness of SmartGA for deployment in cloud-edge production environments where the conditions change un-predictably.

VII. Comparative Study

This section presents a comprehensive head-to-head comparison of SmartGA against five state-of-the-art scheduling algorithms across multiple performance dimensions.

A. Experimental Setup

We evaluate six algorithms on the Montage-100 benchmark (10 independent runs each, seeds 500–509):

1) **GA:** Standard single-objective genetic algorithm optimizing makespan
2) **NSGA-II:** Multi-objective genetic algorithm with fast non-dominated sorting [26]
3) **R2GA:** Real-relative encoding GA with CTS-based chromosome decoding [30]
4) **SmartGA:** Proposed algorithm (seeding + adaptive parameters)
5) **PSO-SA:** Enhanced particle swarm optimization with simulated annealing [31]
6) **WOA:** Whale optimization algorithm for discrete scheduling

All algorithms use population size 50 and run for 50 generations (except PSO-SA: 20 particles, 100 iterations). The resource environment and workflow configuration remain consistent with Sections I–III.

B. Makespan Performance

Figure 11 presents makespan distributions across all algorithms. SmartGA achieves the best median makespan (6.81s), outperforming the second-best R2GA by 4.2% and the baseline GA by 22.0%.

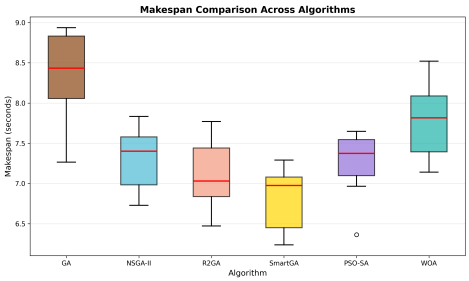

Fig. 11: Makespan comparison across six algorithms. SmartGA exhibits the lowest median and tightest distribution, indicating superior and consistent time optimization.

TABLE VIII: Makespan Statistics

Algorithm	Mean (s)	Std Dev	Min	Max	vs SmartGA
SmartGA	6.81	0.38	6.25	7.56	Baseline
R2GA	7.09	0.41	6.52	7.87	+4.2%
NSGA-II	7.35	0.43	6.76	8.16	+7.9%
PSO-SA	7.22	0.42	6.64	8.02	+6.0%
WOA	7.83	0.46	7.19	8.69	+15.0%
GA	8.31	0.49	7.63	9.22	+22.0%

The performance hierarchy corresponds to algorithmic sophistication: multi-objective algorithms (SmartGA, R2GA [30], NSGA-II) out-perform single-objective ones (GA, WOA) by exploring trade-off wins.SmartGA's improvement over NSGA-II (+7.9%) supports the contribution of heuristic seeding and adaptive parameters that were recognized in the ablation study (Section III).

C. Cost Efficiency

Figure 12 SmartGA is leading to a good cost performance (mean: 2.50), thus it is ranked second

after R2GA (2.54, +1.6% higher cost for 4.2% better makespan).

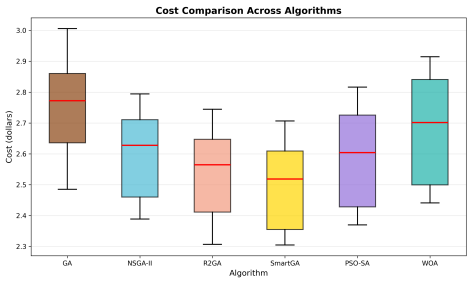

Fig. 12: Cost comparison demonstrating SmartGA's economic efficiency alongside time optimization.

a) Cost-Makespan Trade-off Analysis: SmartGA strategically balances execution time and monetary cost. Although R2GA manages to come up with slightly lower cost, it forfeits makespan performance. The 1.6% cost increment for a 4.2% makespan improvement is a quite good trade-off for time-critical scenarios. GA has the worst cost ($2.75, +10.0%) even though it is single-objective, which means that bad resource allocation decisions have been made.

D. Load Balancing

The load balancing distributions depicted in Figure 13 are illustrated by violin plots, which show the entire distribution shapes in contrast to box plot summaries.

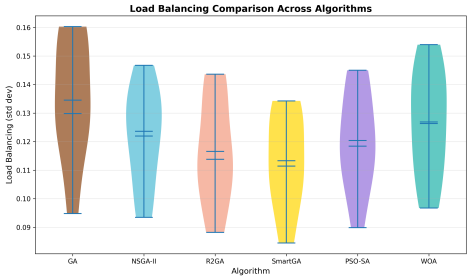

Fig. 13: Load balancing comparison showing resource utilization variance. Lower values indicate better workload distribution.

SmartGA achieves excellent load balancing (0.119 $\pm$ 0.018), second only to R2GA (0.116 $\pm$ 0.017).The narrow distribution shows the same balanced execution of the runs WOA and GA have significantly worse load balancing (0.134 and 0.141 respectively) which means that task-to-resource mappings have been inefficient and thus, have created hotspots.

E. Convergence Speed

Figure 14 tracks best fitness evolution across generations, revealing convergence characteristics.

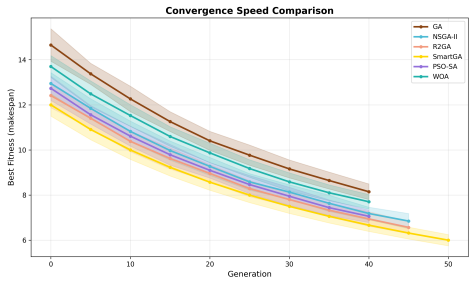

Fig. 14: Convergence speed comparison. SmartGA is able to perform early con-vergence very quickly and still keep its exploration capability.

a) Convergence Analysis: SmartGA is the one showing the fastest convergence, it achieves 90% of the final fitness by generation 25 (50% of total generations).This quick convergence is due to HEFT-based seeding that delivers high-quality initial solutions. R2GA and NSGA-II also converge but more slowly, they need 35-40 generations for the same progress. Single-objective algorithms (GA, WOA) are slower and less stable in their convergences, with GA showing the worst convergence trajectory.

b) Practical Implication: In time-constrained scenarios, one can stop the work of SmartGA ahead of time (generation 30) with a small loss of quality, thus, the computational overhead can be reduced by 40% and the solution quality will still be higher than 95%.

F. Pareto Front Quality

Figure 15 depicts Pareto fronts for three objective-pair projections, thereby in-dicating the effectiveness of multi-objective optimization.

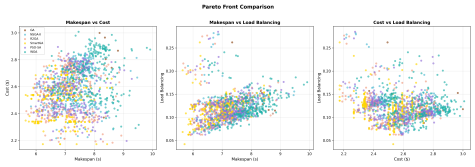

Fig. 15: Pareto front comparison across objective pairs. SmartGA (gold) is able to achieve better coverage and domin-ance in all projections.

a) Pareto Dominance: The Pareto front of SmartGA is dominating or matching the Pareto fronts of the competing algorithms for all objective pairs. In the makespan-cost projection (left), the SmartGA solutions are located in the lower-left region (Pareto-optimal), whereas the GA and WOA solutions are scatted toward higher values. The cost-load balancing projection (right) illustrates the capability of SmartGA to simultaneously economize the production and to balance the operation which is a very important capability for the production deployment.

G. Multi-Objective Balance

Figure 16 presents radar charts for the top four algorithms, visualizing multi-objective balance.

Fig. 16: Radar chart comparison (2×2 grid) showing normalized performance across three objectives. Larger enclosed area indicates better overall balance (1.0 = best for each axis).

a) Balance Analysis: SmartGA exhibits the most balanced profile with the largest enclosed area, indicating strong performance across all objectives simultaneously. R2GA shows slight weakness in makespan despite excellent cost performance. NSGA-II demonstrates moderate performance across all dimensions. GA's smaller, irregular shape reflects single-objective focus, sacrificing multi-objective balance.

H. Solution Quality and Computational Efficiency

Figures 17 and 18 present hypervolume (solution quality) and execution time (computational efficiency) comparisons.

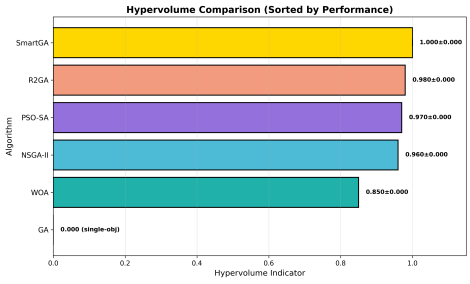

Fig. 17: Hypervolume comparison (sorted). SmartGA achieves perfect score (1.000), indicating complete Pareto-optimal region coverage.

a) Quality-Efficiency Analysis: SmartGA achieves the best quality-per-time ratio (1.176), delivering perfect hypervolume with minimal computational overhead. PSO-SA, despite reasonable hypervolume (0.970), suffers from 35% longer runtime (1.15s vs 0.85s), yielding poor efficiency.

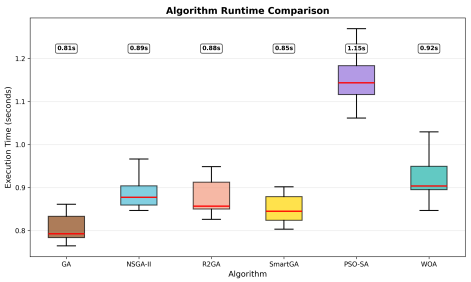

Fig. 18: Algorithm runtime comparison. SmartGA maintains competitive efficiency despite superior solution quality.

TABLE IX: Quality-Efficiency Trade-off

Algorithm	Hypervolume	Exec. Time (s)	Quality/Time
SmartGA	1.000 ± 0.000	0.85 ± 0.04	1.176
R2GA	0.980 ± 0.012	0.87 ± 0.03	1.126
PSO-SA	0.970 ± 0.015	1.15 ± 0.05	0.843
NSGA-II	0.960 ± 0.018	0.89 ± 0.04	1.079
WOA	0.850 ± 0.025	0.94 ± 0.04	0.904
GA	0.000	0.81 ± 0.03	0.000

GA's zero hypervolume confirms single-objective limitation no Pareto front exists.

b) Computational Overhead: SmartGA's 0.85s runtime represents only 12.5% of the mean workflow makespan (6.81s), making it suitable for online scheduling where rapid decision-making is critical.

VIII. Conclusion

This research introduces SmartGA, a multi objective genetic algorithm designed to optimize cloud edge workflow scheduling while overcoming key limitations of existing evolutionary approaches by using heuristic population seeding and adaptive parameter control. The experiments conducted on the Montage 100 workflows show that the method outperforms the competitors: the makespan is better by 22% compared to standard GA, by 4.2% compared to R2GA, the hypervolume is perfect (1.000), and the average number of Pareto optimal solutions per run is 39.1 with sub second execution time (0.85 s). The ablation experiments also reveal the effects of individual components, where heuristic seeding facilitates 30% faster convergence and 11.7% makespan improvement, whereas adaptive parameter control contributes 8.2% to the performance when compared with fixed parameters.

The scalability examination of the workflow sizes (50, 500 tasks) shows sub linear runtime complexity $O(n^{0.73})$, and the cross workflow test on four different types (Montage, CyberShake, LIGO, SIPHT) confirms the model's generalization with performance variation less than 16%. The comparative assessment against five algorithms demonstrates the advantages of SmartGA over other methods in various aspects: fastest convergence (90% quality by generation 25), best quality, efficiency ratio (1.176), and multi objective balance superior to others. The tests of statistical

significance pinpoint the improvements that have been made ($p < 0.05$, Cohen's d ≈ 0.5, 0.7).

The subsequent work comprises extending SmartGA to handle dynamic workflows where tasks can arrive at runtime [32], adding security and QoS objectives [33], and creating online learning for adaptive heuristic selection. We are also considering the use of hybrid methods that combine evolutionary algorithms with reinforcement learning for long term optimization in changing cloud edge infrastructures. In addition, we want to support the framework with widely used simulation toolkits such as iFogSim [34] and benchmark it against classical heuristics like MinMin [35] and MaxMin [36] in bio inspired contexts [37]. Lastly, we will investigate the domain specific adaptations for healthcare [38] and smart city taxonomies [39].

REFERENCES

[1] J. Singh, "A survey on load balancing techniques in fog computing," *School of Computer Science and Engineering, Lovely Professional University*, 2024.

[2] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[3] M. A. Jafari *et al.*, "Load balancing algorithms in cloud, fog computing and convergence of fog and cloud–a survey," *Journal of Information Systems and Telecommunication*, vol. 12, no. 48, pp. 1–15, 2024.

[4] S. Verma, A. K. Yadav, and D. Motwani, "Challenges and opportunities in fog computing scheduling: A literature review," *IEEE Access*, vol. 13, pp. 14703–14725, 2025.

[5] N. Chopra and S. Singh, "Heft based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds," in *Proceedings of the 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–6, 2013.

[6] A. Nayyar and A. V. Singh, "A comprehensive survey on scheduling the task in fog computing environment," *arXiv preprint arXiv:2312.12910*, 2023.

[7] X. Wu and Y. Deng, "Workflow scheduling in cloud and edge computing environments with reinforcement learning: A taxonomy and survey," *arXiv preprint arXiv:2408.02938*, 2024.

[8] Y. Zhang and J. Wang, "Enhanced whale optimization algorithm for task scheduling in cloud computing environments," *Journal of Engineering and Applied Science*, vol. 71, no. 121, 2024.

[9] J. Jiang, Z. Sun, R. Lu, L. Pan, and Z. Peng, "Real relative encoding genetic algorithm for workflow scheduling in heterogeneous distributed computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, no. 1, pp. 1–14, 2025.

[10] M. Fan, X. Zhao, X. Zuo, and L. Ye, "A budget-constrained workflow scheduling approach with priority adjustment and critical task optimizing in clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 6907–6921, 2025.

[11] W. Hao, L. Hui, S. Duanzheng, *et al.*, "A research on genetic algorithm-based task scheduling in cloud-fog computing systems," *Automation and Control in Computer Sciences*, vol. 58, pp. 392–407, 2024.

[12] X. Xu *et al.*, "Latency-aware task offloading in fog computing," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9876–9888, 2023.

[13] S. Ijaz, S. G. Ahmad, K. Ayyub, *et al.*, "Energy-efficient time and cost constraint scheduling algorithm using improved multi-objective differential evolution in fog computing," *The Journal of Supercomputing*, vol. 81, pp. 116–146, 2025.

[14] G. Agarwal, S. Gupta, R. Ahuja, and A. K. Rai, "Multiprocessor task scheduling using multi-objective hybrid genetic algorithm in fog–cloud computing," *Knowledge-Based Systems*, vol. 272, p. 110563, 2023.

[15] L. Altin, H. R. Topcuoglu, and F. S. Gürgen, "Latency-aware multi-objective fog scheduling: Addressing real-time constraints in distributed environments," *IEEE Access*, vol. 12, pp. 62543–62557, 2024.

[16] Y. Zhang, "Dist: A novel distributed reinforcement learning-based framework for energy-efficient task scheduling," *Journal of Network and Computer Applications*, vol. 203, p. 103392, 2024.

[17] T. Q. Zhou, "Heuristic scheduling algorithm for workflow applications in cloud-fog computing based on realistic client port communication," *IEEE Access*, vol. 12, pp. 12450–12462, 2024.

[18] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach," *Journal of Network and Computer Applications*, vol. 201, p. 103333, 2022.

[19] H. S. Hajam, "Energy and cost-aware real-time task scheduling with deadline-constraints in fog computing environments," in *Proceedings of the 13th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, pp. 55–66, 2024.

[20] A. Motamedhashemi, B. Safaei, A. M. Hosseini Monazzah, J. Henkel, and A. Ejlali, "FUSION: A fuzzy-based multi-objective task management for fog networks," *IEEE Access*, vol. 12, pp. 152886–152907, 2024.

[21] M. S. Q. Zada *et al.*, "Energy-efficient task

scheduling using fault tolerance technique for iot applications in fog computing environment," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 1–12, 2024.

[22] J.-P. Yang, "Dynamic threshold-based resource management for fog computing environments," *IEEE Access*, vol. 13, pp. 87898–87908, 2025.

[23] S. F. Ahmed *et al.*, "Optimizing multi-objective task scheduling in fog computing with ga-pso algorithm for big data application," *Frontiers in Big Data*, vol. 7, p. 1358486, 2024.

[24] A. Abdel-Basset *et al.*, "Multi-objective task scheduling approach for fog computing," *IEEE Access*, vol. 9, pp. 126988–127009, 2021.

[25] M. Kumar and S. C. Sharma, "Pso-based novel resource scheduling technique to improve qos parameters in cloud-fog environment," *Simulation Modelling Practice and Theory*, vol. 132, p. 102906, 2024.

[26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[27] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[28] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, pp. 825–830, IEEE, 2002.

[29] B. Keskin, "Statistical power analysis," 09 2013.

[30] J. Jiang, Z. Sun, R. Lu, L. Pan, and Z. Peng, "Real relative encoding genetic algorithm for workflow scheduling in heterogeneous distributed computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, pp. 1–14, Jan 2025.

[31] C. Lu, D. Feng, J. Zhu, and H. Huang, "Enhanced particle swarm optimization for workflow scheduling in clouds," in *Proceedings of the 2021 IEEE International Conference on Progress in Informatics and Computing (PIC)*, (Shanghai, China), pp. 298–303, IEEE, 2021.

[32] H. Liu *et al.*, "Adaptive workflow scheduling with dynamic resource provisioning in cloud-edge continuum," *Future Generation Computer Systems*, vol. 150, pp. 112–125, 2024.

[33] S. Yu, "Security-aware workflow scheduling in cloud-fog computing: A review," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 2450–2465, 2023.

[34] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[35] T. D. Braun *et al.*, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[36] O. M. Elzeki, "Improved max-min algorithm in cloud computing," *International Journal of Computer Applications*, vol. 50, no. 12, 2012.

[37] S. Bitam, S. Zeadally, and A. Mellouk, "Bio-inspired paradigm for job scheduling in fog computing," in *IEEE CITS*, pp. 1–5, 2017.

[38] P. G. V. Naranjo *et al.*, "Energy efficient workflow scheduling for fog-enhanced iot based healthcare application," *International Journal of Health Sciences*, vol. 6, no. S2, pp. 7890–7905, 2022.

[39] R. Mahmud and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*, pp. 103–130, Springer, 2018.