National College of Ireland

Improving Task Scheduling and Resource Allocation by Optimizing Swarm Based Metaheuristic Algorithm with Evolutionary Based Optimizer

MSc Research Project
Masters in Cloud Computing

Priya Bisht
Student ID: X23109394

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Priya Bisht
Student ID:	X23109394
Programme:	Masters in Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Diego Lugones
Submission Due Date:	05/09/2024
Project Title:	Improving Task Scheduling and Resource Allocation by Optimizing Swarm Based Metaheuristic Algorithm with Evolutionary Based Optimizer
Word Count:	XXX
Page Count:	23

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:	15th September 2024

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):	

Improving Task Scheduling and Resource Allocation by Optimizing Swarm Based Metaheuristic Algorithm with Evolutionary Based Optimizer

Priya Bisht
X23109394

Abstract

Cloud computing has transformed traditional service deployment methods, allowing users to access various services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) without investing in their own infrastructure. Task scheduling within these environments is critical for optimizing resource utilization, minimizing completion times, and ensuring cost-effectiveness. Scheduling tasks with varying complexities and resource requirements remains a significant challenge due to its NP-complete nature. This research addresses this challenge by proposing a hybridized approach that integrates swarm-based and evolutionary-based metaheuristic algorithms to enhance task scheduling efficiency. The research brings forward a combined modal which focuses on optimizing the Honey Badger Algorithm (swarm based) with Genetic Algorithm (evolutionary based) to improve task scheduling performance by reducing makespan time. this is compared to AntLion Optimizer and original HBA. With such a combination of global exploration capabilities of swarm based with refinement and exploitation strengths of evolutionary algorihms the proposed optimized algorithm tries to achieve better allocation of tasks and resource utilization with improved makespan time. This optimised algorithm was tested against the Ant Lion Optimizer and the original HBA through multiple scenarios with varying values of Probability ratio (PR) and VM and task counts. The analysis also implies the importance of right choice of algorithm based on the specific task requirements, considering the number of VM, tasks, and varying hyperparameters like PR in this case. Analysis results show that the proposed hybrid HBA-GA approach outperforms existing methods in terms of reducing makespan time and balancing workload distribution across resources thus increasing performance. This research contributes to the advancement of cloud computing task scheduling by presenting a novel hybrid algorithm that enhances scalability, reliability, and overall performance, addressing the limitations of traditional and standalone metaheuristic algorithms.

1 Introduction

Cloud Computing is a growing area within distributed computing utilised for various applications such as data analysis, storage etc. This technology has revolutionized traditional methods of service deployment for both businesses and individuals by offering different types of services over internet. Users can access these services without the need

to invest in their own computing infrastructure and can choose from difference services provided by providers such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). To use these services, users submit requests over the Internet to the cloud providers, who then manages the necessary resources. Each of these request is treated as a task and Cloud providers use scheduling algorithms to efficiently allocate resource and handle the tasks requested by users. Keivani and Tapamo (2019)

1.1 Research Background/ Objective

Task scheduling is a very crucial component as since it is incharge of determining the sequence of tasks execution with effective resources utilisation of computations while maintaining application performance with cost-effectiveness. All the tasks can have varying complexities starting from simple data processing jobs to complex scientific computations and each of these will have their own resource requirements. Houssein et al. (2021) The main problem with cloud providers when it comes to scheduling varied number of tasks with varied complexities on defined resources is to keep scheduling time of completion as per the delivery time objectives set by users application objective. "*The scheduling algorithms should not take longer to search and allocate tasks to resources with efficient scheduling and distributing the tasks efficiently to all the existing resources in such a way that no single resource is over utilised*" which is *problem objective* in this research. In this research I aim to improve the actual time of scheduling tasks which is crucial and important since delays can lead to resources getting underutilized and increase the costs of reserving the resources on cloud thus, impacting the efficiency of the whole system. Efficient task scheduling makes sure that all the tasks are scheduled within an "optimal timeframe" thus reducing the the overall execution time (makespan).

Task scheduling is considered as a combinational optimisation problem in which standard algorithms fail to identify the global optimal solution. It is regarded as an NP-complete problem because of the exponential number of possible task-to-VM assignments. In this research, the objective function works towards maximizing the minimum load across all VM's to guarntee balanced workloads. With M number of tasks and N number of VMs, there are N^M possible ways. This exponential rise makes it computationally complicated to find an optimal solution which is a chief and prominent characteristic of NP-complete problems. To give an example inside my research with 60 number of VMs and 375 number of tasks there are 60^{375} possible ways to assign these tasks thereby showcasing the challenge of assessing all the possible configurations. This property of the problem objective makes it as an NP-complete problem.

In Singh et al. (2020)the total time required to complete a defined sequence of tasks should not be longer than expected which can increase costs and can lead to system over utilisation. Here scheduling tasks is a single-objective and an optimal scheduling algorithm should provide efficient solutions that the problem objective is satisfied. It is very important to efficiently handle parallelism, large scale scheduling and minimize this makespan time of tasks scheduling and resource allocation for a maximized resource utilization and on time task allocation and completion. The main objective is to obtain optimal value for our goal. The purpose of scheduling is to schedule tasks that pass through a routes and policies to optimize total completion time or makespan and tasks are allocated that no single resource is over utilised. Bittencourt et al. (2018) Traditional scheduling techniques includes methods like First-Come-First Serve or Round Robin scheduling are

not able to solve the the scheduling problem with good accuracy because of the inherent complexity and are inefficient for the dynamic changes in resources or different priorities of tasks resulting in suboptimal performance. There are innovative search algorithms Heuristic and metaheuristic algorithms that have been proposed individually or in hybridised form to solve these complex problems and find near-optimal solution. Heuristic algorithms such as minimum execution time (MET), minimum completion time (MCT), shortest job to fastest processor (SJFP), longest job to fastest processor (LJFP), Min-Min, and Max-Min are used for static scheduling tasks. This problem is classified as a non-deterministic polynomial (NP)-hard optimization problem. As a result, metaheuristic algorithms are nature based algorithms that are effective for tackling these complex multi modal optimization problems. Singh et al. (2020) Metaheuristics can be used in different types of problems as they showcase a class of generic search algorithms. Singh et al. (2021) Metaheuristic are of four types: human based, physics based, evolution based, and swarm based algorithms. The main metaheuristic algorithms that have been implemented for task scheduling are ALO ,Abualigah and Diabat (2021) ACO Mahato et al. (2017)and PSO Alsaidy et al. (2022)and are swarm based methauristic algorithms.. These algorithms are inspired by the collective behavior of social insect colonies, where individual agents collaborate and coordinate to solve complex problems. Algorithms such as Ant Lion Optimization (ALO) or Honey Badger Algorithm (HBA) model these swarm behaviours to discover optimal solutions.

These swarm based algorithms can be optimised using evolutionary based algorithms. Evolutionary algorithms principle of natural selection, genetics and employing genetic operators like selection, crossover, and mutation to iteratively evolve a population of candidate solutions. Each solution fitness is assessed based on predetermined objectives, with fitter individuals being more likely to reproduce and generate future generations of potentially better solutions. This iterative process enables evolutionary algorithms to explore the search space and converge on optimal or near-optimal solutions. Several research have explored these techniques as viable solutions for efficient task scheduling solving the problem objective.

Both evolutionary and swarm intelligence techniques have been effectively applied to various optimization challenges, including task scheduling on identical parallel processors. These methods offer advantages like flexibility, adaptability, and the capability to manage complex optimization environments, though their effectiveness can vary depending on the specific problem objectives and its characteristics.

1.2 Research Question

This research propose a hybridised approach that focus on hybridizing swarm-based algorithms with evolutionary-based algorithms as this can harness the strengths of both approaches: the global exploration and adaptive behavior of swarm algorithms, and the effective exploitation and refinement capabilities of evolutionary algorithms can enhance the solution diversity and convergence speed, making it useful for solving defined problem objective. This gives the research question "***How can the makespan time be improved for task scheduling and resource allocation in cloud by optimizing swarm based metaheuristic algorithm with evolutionary based optimizer ?***". This research proposes an optimized Honey Badger Algorithm with Genetic Algorithm for the defined problem objective. Research will explore how hybridisation of swarm based and evolutionary based Metaheuristic search algorithm can improve in terms of makespan

for solving the defined problem objective. This research also explores how varying values of PR and problem scenarios of different VM and task counts affect the performance of the proposed hybridised algorithm.

1.3 Motivation

The motivation behind this research origins from the ever increasing complexity and demand for efficient task scheduling and resource allocation in cloud. Ancient and traditional scheduling methods like First-Come-First-Serve and Round Robin are increasingly inadequate due to their inability to take control the dynamic changes in resources and varying task priorities. The proposed research seeks to address these challenges by hybridizing swarm-based algorithms with evolutionary-based algorithms, aiming to leverage the global exploration capabilities of swarm intelligence and the robust optimization properties of evolutionary strategies. This study will explore the hybridisation of the Honey Badger Algorithm with Genetic Algorithm, targeting the reduction of makespan time and enhancing resource utilization and solving the problem objective. This hybrid approach is expected to provide novel approach for improved task scheduling and reliable solutions for the defined research problem objective.

1.4 Report Structure

Existing literature on task scheduling and gaps addressed by this study is presented in the Related Work section. The Methodology section outlines the problem formulation, tools, and algorithms used, focusing on the hybridization of the Honey Badger Algorithm and Genetic Algorithm. The Design Specification details the system architecture and workflow, the Implementation section covers practical aspects of integrating and deploying the solution. The Evaluation section analyzes experimental results, comparing the proposed approach's effectiveness by comparing it to the existing promising algorithms in reducing makespan and improving resource utilization. The Conclusion and Future Work section summarizes key findings, discusses limitations, and suggests areas for further research, emphasizing the study's contributions to enhancing cloud system efficiency and reliability.

2 Related Work

The emergence of cloud computing has turned the workings of various organizations, how they manage and deploy the resources needed for computations. With this growth it has also brought forward numerous distinctive challenges for cloud service providers for taking these requests by the users as a task and assigning these tasks efficiently to the resources. It is very important to have efficient task scheduling since it directly impacts the operations and resource utilization inside multiple cloud environments.

2.1 Challanges in Task Scheduling and Evaluating Algorithms: Heuristic and Metaheuristic

The research conducted by Houssein et al. (2021) identifies a number of issues with task scheduling like underutilization and overutilization resulting in wastage of computing resources or degraded performance. The scheduling problems are NP-hard stating that

they require large computational time to find near-optimal solution making it very difficult to implement and main goal being optimizing various QoS parameters like cost, response time, and energy consumption which is a significant challenge and needs to be addressed for development of any metaheuristic algorithm. This paper provides a review of meta-heuristic approaches and categorizes them on their nature, objectives, task-resource mapping to understand the complexity of the algorithms and then compares them on various metrics reviewing various simulation tools and providing insights. Ghafari et al. (2022)also presents a comparative analysis of scheduling methods aimed at minimizing energy consumption and broadly categorized them into three main groups namely heuristic-based, meta-heuristic-based, and other task scheduling algorithms highlighting benefits and limitations to understand the trade-offs involved in choosing a method, making this paper valuable to select the most suitable algorithm to serve their needs. Heuristic-Based Task Scheduling use rule-based methods and logic to schedule tasks and are less intensive but not always provide the optimal solution with dynamic environments. The Meta-heuristic are more flexible and capable of handling complex scheduling problems including hybrid algorithms not strictly falling into the heuristic or meta-heuristic categories. Limitations of the above literature reviews being the above analysis limited to few algorithms and not covering the rapid evolution of cloud and new algorithms.

2.2 Comparitive analysis of Single Metaheuristic Algorithms

The work by Mandal and Acharyya (2015) also brings to light several challenges and provides an implementation of three meta-heuristic algorithm namely Simulated Annealing, Firefly Algorithm, and Cuckoo Search Algorithm tested on a GCC compiler and compared based on cost and execution time with focus on minimizing processing time. It demonstrates firefly algorithm outperforming SA and Cuckoo Search Algorithm but lacks the Differential Evolution or Hybrid Algorithms. Secondly the initial solutions were generated randomly, and lacked scalability. Thirdly the parameter tuning method is not explicitly defined that can influence the performance. Similar attempt for a comparative analysis and review has been made by Singh et al. (2021) on six prominent metaheuristic algorithms Ant Colony Optimization, Particle Swarm Optimization, Genetic Algorithm, Artificial Bee Colony, Crow Search Algorithm, and Penguin Swarm Optimization and have been focused using makespan and utilization cost. All these were conducted on cloudSim. The results provided Crow Search Algorithm as the most optimal technique for minimizing makespan and cost. Other promising techniques have not been included and does not capture all the complexities associated. It has no mention of energy consumption and hybrid directions or development of more sophisticated models to better handle dynamic and heterogeneous cloud environments. The metaheuristics have been leveraged by Keshanchi et al. (2017)by formal verification, simulation and statistical testing using priority queues. An improved genetic algorithm model for task scheduling using Linear Temporal Logic and verification tools like PAT and NuSMV for deadlock freedom has been proposed. A powerful one-point crossover operator and elitism technique for maintaining precedence constraints with reduced chances of generating invalid solutions has also been propsed. The work by Prity et al. (2023) proposes an algorithm for scheduling based on bacterial foraging optimization (BFO) minimizing the idle time of virtual machine and ensuring load balancing with reduction in runtime and energy by mapping tasks to cloudlets using BFO and assessing it on CloudSim. BFO based algorithm provides solution for cloudlet scheduling and outperformed algorithms like SOS,

EATSGA, GAACO in achieving a lower execution time with better energy consumption and allocation of resources. But both the works above needs improvements to handle highly dynamic and unpredictable environments and faced increasing complexity and scalability constraints inspite of reductions in the execution time.

2.3 Comparitive analysis of Improved or Modified Metaheuristic Algorithms

Based on the work by Chen et al. (2020) an Improved Whale Optimization Algorithm (WOA) is proposed where the algorithm worked on minimizing task execution time, system load, and cost respectively. It successfully addresses these objectives simultaneously and uses nonlinear convergence factor and dynamic population control. It outperforms Ant Colony Optimization, Particle Swarm Optimization in all scenarios and reduces the total by more than 20percent along with an improved convergence speed. The same has been addressed by Guo et al. (2012)where Particle Swarm Optimization has been tailored and enhanced with a small position value rule to convert continuous position values into discrete task assignments with much better convergence and execution times that CM-PSO and L-PSO and has better scalability with large numbers of processors and tasks with lower costs. While both of them showed results and contributions, but their implementation complexity is a barrier. They have increased computational overhead with the advanced optimization strategies that can limit its applicability. They also have scalability issues with unexplored parallel implementations. Optimizing QoS problems by prioritizing tasks with higher importance and devaloping a highly adaptive system is also a challenge. Furthermore, the implementation of the PSO requires tuning of parameters like inertia weight, and social coefficient that is time-consuming.

2.4 Exploring Hybrid and Adaptive stratergies of Metaheuristic Algorithms

Research by Mirjalili (2015) where AntLion optimizer has been devised mimicking the foraging behaviour of antlion larvae and involving steps from random walks of ants to rebuilding traps have been depicted. This ensures a dynamic balance between exploration and exploitation with the help of adaptive shrinking of the search boundaries preserving the best solutions. The paper shows its benchmarking across three phases where a set of 19 mathematical functions were used to evaluate its performance on different characteristic thus demonstrating superior performance in exploration, local optima avoidance, and convergence. The Engineering Problems where it was applied to provided near-optimal solutions thereby showing its applicability to constrained problems. Kakkottakath Valappil Thekkepuryil et al. (2021) proposes this ALO hybridization with PSO to address the existing issue of task scheduling. The proposed hybrid ALO-PSO algorithm showed improvements like costs by 9.8percent compared to GA-PSO, 10percent to PSO, 20percent to ALO, 30percent to RR. Significant reductions in load balancing and makespan like 8percent compared to GA-PSO, 10percent compared to ALO, 20percent compared to PSO, 35percent compared to RR, and 45percent compared to GA were also seen. Abualigah and Diabat (2021) introduces another hybrid optimization version called Multi-Objective Hybrid Antlion Optimizer combining ALO with elite-based Differential Evolution where it had superior performance compared to GA, PSO, and original ALO. It used synthetic and real trace datasets for demonstrating its superior

performance confirmed by Statistical t-tests. This MALO showed minimal improvement in makespan values for specific task sizes and hyper parameters. Thus, all these works concluded the better performance of ALO individually and when hybridised with another metaheursitc algorithm as well highlighting a few limitations as well. But apart from all these results the limitations encountered in ALO papers here were the complexity due to the hybrid nature and sensitivity of parameters requiring tuning thus a need to explore adaptive parameter tuning techniques. High computational running time, susceptibility to yielding local optimum solutions and imbalance between local and global optima.

Manikandan et al. (2022) brings another adaptive hybrid optimization approach where Whale Optimization Algorithm is combined with the Bees Algorithm calling it Hybrid Whale Optimization Algorithm-based MBA (HWOA-MBA) making use of both their strengths. The Random Double Adaptive Whale Optimization Algorithm (RDWOA) is intensified with a mutation operator from the Bees Algorithm for making the solution quality and convergence speed better and was evaluated on cloudsim giving significant improvements aswell. Inspite of these results the approaches above targets compute intensive and independent tasks and thus limits applicability. Also, in the other research the combination of two algorithms increased the complexity of integration and thus required more resources

2.5 Analyizing Honey Badger and Genetic Algorithm

Hashim et al. (2022)introduced a Honey Badger Algorithm inspired by its foraging behaviour divided into two main phases: the "digging" and the "honey phase," corresponding to exploration and exploitation. It makes use of a combination of randomization and a decreasing factor to maintain population diversity and avoid premature convergence. The paper showed it to outperform ten well-known metaheuristic algorithms in convergence speed and solution quality on 24 benchmarks since its parameters are easily adjustable and can be tailored for different optimization problems. However, it alone has a challenge of sensitivity to parameters which can hamper its performance and computational complexity with increased costs.

Genetic Algorithm Task Allocation approaches are utilized by Rekha and Dakshayini (2019) where a model allocates tasks to virtual machines based on their processing capabilities and the size of the tasks. The fitness function evaluates the efficiency of task allocation, considering execution and resource allocation time. This approach reduces the makespan also gives better throughput. Similarly, Pirozmand et al. (2021) proposes integrating Genetic Algorithm with Energy-Conscious Scheduling heuristic algorithm involving task prioritiy using GA to generate initial chromosomes, followed by task assignment to processors using an energy-conscious heuristic. It addresses both makespan and energy consumption scalability on different sets of tasks from 30 to 300 tasks and different hyper parameters outperforming GSA, ABC, DA, and various PSO variants. Agarwal and Srivastava (2016) also proposes the same Genetic Algorithm to optimize workloads and compares it to Greedy-based and First-Come-First-Serve using CloudSim. Its objective function lays stress on minimizing the execution time of tasks by assigning them to particular VMs, ensuring that the maximum time taken is minimized. Thus, the GA approach outperformed FCFS and Greedy algorithms in execution time and handled both single-objective and multi-objective optimization problems. The GA approach taking 100.55 time units to execute whereas FCFS takes 343.52 time units and Greedy Based strategy took 101.79 time units respectively. The common limitations of all

the works include computationally intensive, highly dependent on the quality of the initial population and initial parameters set. Thus from the above review Genetic algorithms known for their robust parameter tuning capabilities, could be used to optimize HBA's parameters dynamically during the search process. However, its sensitivity to parameters presents a challenge that could potentially be addressed by integrating genetic algorithms for dynamic parameter optimization.

2.6 Research Niche

However, despite of all these studies the research gap lies in lack of research on optimizing the Honey Badger Algorithm with Genetic algorithm for reducing the make span time. It would be interesting to compare against Antlion optimizer(ALO) which has proved to be an optimal algorithm for similar problem objective. The choice of ALO has been made after reviewing a couple of papers that make use of metaheuristic algorithms in task scheduling and resource allocation as shown in the literature review above. A few highly cited papers show ALO is robust, has high exploration and faster convergence speed compared to other popular and common meta-heuristic algorithms like Particle Swarm Optimization, Bat Algorithm, Genetic Algorithm, Whale Optimization Algorithm etc. ALO also has showcased good global search abilities and has been very efficient in handling similar complex problems as my implemented research objective with wide search spaces and is particularly suitable for optimization, task scheduling and engineering problems.

Also from above reviewed papers it is observed that the experimentations for each algorithm has been performed either in specific ranges of task sizes by varying hyper parameter configurations but not specifically addressed to varying the PR (i.e probability ratio) levels that controls the probability of random behavior or mutation within the algorithm's optimization process. While many hybrid approaches exist, optimizing HBA with GA remains uninvestigated to increase the task scheduling performance by leveraging the strengths of both. Addressing such a gap could potentially contribute to more efficient task scheduling and management of resources in cloud.

3 Methodology

This research focus on hybridizing swarm-based algorithms with evolutionary-based algorithms. This research analyses the Task scheduling on identical parallel machines using a comparative evaluation of swarm based meta-heuristic algorithms, such as Honey Badger Algorithm (HBA), Ant Lion Optimizer(ALO) and the proposed hybridisation of swarm based Honey Badger Algorithm (HBA) with evolutionary based genetic algorithm (GA) with the objective to design a fitness function that minimizes the makespan and considers the satisfaction of the research problem objective. This problem objective defines the real world problem scenario of Task scheduling in Cloud Service Provider back end.

3.1 Problem Formulation

The rapid growth of cloud computing has revolutionized service deployment, enabling users to access a variety of services without investing in their own infrastructure. However, this flexibility brings the challenge of efficiently managing and scheduling diverse user requests, or tasks, within a cloud service provider's datacenter. These tasks, varying in

complexity and resource requirements, must be scheduled and executed in a manner that meets user-defined delivery time objectives while optimizing resource utilization. The core problem lies in developing a scheduling algorithm capable of efficiently allocating these tasks to available resources, minimizing scheduling delays, preventing resource over-utilization. Given the combinatorial nature of task scheduling, it is classified as an NP-complete problem, making it crucial to devise an algorithm that can provide near-optimal solutions within acceptable time frames. This research aims to address these challenges by formulating and solving the task scheduling problem in cloud computing environments.

3.1.1 Objective Function/Problem Objective

This objective function aims to define the research problem objective that is to allocate the tasks to the defined number of VMs so that all workloads are evenly distributed to achieve an overall balanced system for optimal performance. By returning the minimum workload of the VMs, it try to maximize the minimum workload. The workload for each VM is calculated by adding the normalized task size (task size divided by VM capacity) to the workload of the corresponding VM. This helps in handling VMs with different capacities. By always returning the minimum workload, the function ensures that no VM is underloaded aiming to avoid scenarios where some VMs are overburdened while others are idle. This is the objective function that is to be optimized. The 'vmcount' and 'taskcount' are provided as inputs to the metaheuristic algorithm containing an objective function.

3.2 Algorithms in consideration

The HBA, GA and ALO have been used to find the best task allocation by iteratively improving the candidate solutions. The solution vector's search space is defined by the bounds parameter as [0, vmcount] for each task meaning that each task can be assigned to any VMs so that the lower bound for each task assignment is 0 while for the upper bound for each task assignment is the total number of VMs (vmcount). The 'minmax' parameter specifies the optimization direction. The Objective function tries to balance the workload. The model is initialised with 'popsize' parameter i.e. the number of candidate solutions considered in each iteration with the algorithm running for a fixed number of iterations (epochs) considering a population of candidate solutions in each iteration. These parameters are varied in the research experimentation to explore behaviour of algorithm solutions.
The optimal solution provide an allocation of tasks to VMs achieving the defined problem objective and research question. Once the the optimization completes, the algorithm solution presents index number as the task number and the value of the index as the VM ID the task is scheduled. The total time taken by the algorithm to provide the solution is calculated and is useful for performance evaluation and answering research question.

3.2.1 The Honey Badger Algorithm

This algorithm is inspired by the behaviour of the honey badgers and their digging and honey phased activities for catching the prey and is a swarm based metaheuristic algorithm. In the paper author Hashim et al. (2022) presents algorithm called Honey Badger Algorithm (HBA) which is based on the working of honey badgers where the parameters like population size (i.e. number of honey badgers involved), Total number of

iterations (Tmax) are defined followed by defining a random position (xi) in the search space from each of them. Next the intensity of prey's scent or its concentration which is denoted by (I) is calculated along with its density factor which makes sure that the algorithm starts from a wider search hence the density factor is highest at the start and narrows down i.e decreases to a more focused search and trying to escape the local optima and reach the global optima with the best solution using a flag (F) which is responsible for changing the direction of search either in cardioid pattern or straight forward pattern and explore new search areas. The fitness functions for each Badger is calculated using an objective function of the optimization problem and if it is better than the previous the position is updated and this iterations keeps on repeating till tmax is reached presented in Hashim et al. (2022)

3.2.2 The Genetic Algorithm

It is inspired from biological genetics and each solution (chromosome) is a way of assigning tasks to VMs. For example, if there are 5 tasks and 3 VMs, a chromosome might look like [VM1, VM2, VM3, VM1, VM2], meaning task 1 goes to VM1, task 2 to VM2, and so on. After this the fitness function is calculated based on balance and makespan time and then the best solutions (with the least makespan time) depending on their fitness scores is selected. Furthermore parts of two parent solutions are combined to generate a new offspring eg taking the first half of one parent and the second half of the other parent. This is called "crossover". Small changes are made to some solutions like moving a task from one VM to another randomly and then replace the old population with new offspring and continue crossover and mutations till best solution is reached as described in Whitley (1994) and Rekha and Dakshayini (2019)

3.3 The AntLion Optimizer

In the paper by Mirjalili (2015) presents Ant Lion Optimizer inspired by hunting techniques of Ant lions. The population of ants (solutions) and antlions (potential solutions) are initialized randomly which corresponds to randomly assigning tasks to VMs and creating an initial task schedule. Fitness (i.e makespan time in this case and lower makespan better fitness) of each ant and antlion is calculated. The best antlion is selected as the elite meaning the best solution found so far (i.e. identifying the task schedule with the lowest makespan). Select an antlion using a roulette wheel mechanism based on fitness i.e giving higher probability to schedules with lower makespan to be selected for producing new schedules thereby avoiding local optima. Then updating the position of the ant by performing a random walk i.e adjusting the task schedule by making small changes influenced by better schedules and gradually decreasing the boundary of the random walk (ensuring both global exploration and local exploitation) to simulate the sliding of ants towards the antlion. Update the position of the ant (task schedule) and calculate its new fitness(makespan). If an ant's fitness (VM task schedule) is better than its corresponding antlion, replace the antlion with the ant and update elite (best task schedule) and return the elite as the best solution found i.e finalize the task schedule with the minimum makespan. Pseudocode for ALO is presented Mirjalili (2015)

4 Design Specification

This design specification outlines a comprehensive experimental approach for optimizing the Honey Badger Metaheuristic Algorithm (HBA) using the Genetic Algorithm (GA) to reduce makespan time in solving the problem objective. The optimized algorithm will be compared with the AntLion Optimizer (ALO) and the base HBA. This research aims to compare the proposed optimized algorithm with the base HBA and ALO to determine if the new proposed approach is more efficient in dynamic cloud computing, specifically in reducing makespan time for the defined problem objective. Below is the step-by-step design specification for this research and the system design is mentioned in Figure 1

1. In this research vm_count, task_count and problem objective is defined and initialised in python notebook.

2. These inputs are given to the algorithm i.e ALO, HBA and HBA Hybridisation with GA (Proposed). These algorithms provide a solution i.e. "which task is allocated to which VM, where, task is the index and the value is the VM_ID while solving the problem objective of this research.

3. After the second step is performed this research record the execution time of the algorithm to provide this solution and this makespan time will be ploted as bar graph for comparison between the ALO, HBA and HBA Hybridisation with GA (Proposed) algorithms.

4. The final result is thoroughly validated and the final results is visualised using MATPLOTLIB bar graph. The similar experiments are performed for all 3 algorithms and are visualized.

5. The above visualization will help me validate if the solution provided by the algorithm is solving my "Problem objective" and answer my "research question".

6. All these experimentation in the research is performed by varying the hyper parameters of the algorithms along with the vm and task counts and performing step 2 to step 7 iteratively. This will answer the research question i.e. "How can the makespan time be improved for task scheduling and resource allocation in cloud by optimizing swarm based metaheuristic algorithm with evolutionary based optimizer.

7. Here the research with the rigorous experimentation will present which hyper parameters configuration of the algorithm will best perform to solve this research Problem objective in with minimum make span time. These experimentation will answer research question and will be able to explain the "how" can this be achieved with more accuracy and less makespan time by exploring the hyperparameter sensitivity of the algorithms.

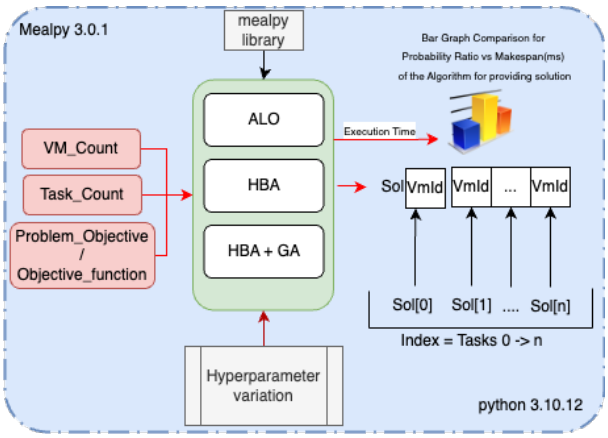

Figure 1: Flow for the execution of algorithms

5 Implementation

The research question focuses on the optimization of Honey Badger with Genetic algorithm to minimize the makespan time and address the task scheduling and resource allocation problem. This implementation of the algorithms has been done using Mealpy python library. The key performance metrics involved is Makespan time which is the total time required to schedule all the tasks on the defined resources solving the research problem objective.

5.1 Implementing Objective Function / Problem Objective

The function `Fun` is developed in python that assigns tasks to virtual machines (VMs) and balances the load across them. This function defines the problem objective of the research.
Let N be the number of VMs.
Let M be the number of tasks.
Let C_i be the capacity of VM i (for i from 1 to N).
Let T_j be the size of task j (for j from 1 to M).
Let x_{ij} be a binary variable that is 1 if task j is assigned to VM i, and 0 otherwise.

The research aim to maximize the minimum workload among all VMs, which can be expressed as maximizing

$$\min(W_1, W_2, \ldots, W_N)$$

where W_i is the workload of VM i.

The workload W_i of VM i is given by:

$$W_i = \sum_{j=1}^{M} x_{ij} \cdot \frac{T_j}{C_i}$$

In the function, all VMs have the same capacity $C_i = 1000$, simplifying the workload calculation to:

$$W_i = \sum_{j=1}^{M} x_{ij} \cdot \frac{T_j}{1000}$$

To maximize the minimum workload, leading to the following optimization problem:

$$\max_{x_{ij}} \left(\min_{i=1,\ldots,N} W_i \right)$$

Substituting the workload expression:

$$\max_{x_{ij}} \left(\min_{i=1,\ldots,N} \sum_{j=1}^{M} x_{ij} \cdot \frac{T_j}{1000} \right)$$

Each task is assigned to exactly one VM:

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \text{for each } j = 1, 2, \ldots, M$$

Binary assignment variables:

$$x_{ij} \in \{0, 1\} \quad \text{for each } i = 1, 2, \ldots, N \text{ and } j = 1, 2, \ldots, M$$

Putting it all together, the mathematical formulation of the problem is:

$$\max_{x_{ij}} \left(\min_{i=1,\ldots,N} \sum_{j=1}^{M} x_{ij} \cdot \frac{T_j}{1000} \right)$$

This distributes the tasks among the VMs in such a way that the minimum workload across all VMs is maximized, resulting in a balanced workload distribution hence defining the problem objective.

5.2 Optimized Honey Badger with Genetic Algorithm

Algorithm 1 Steps for Genetic Algorithm for HBA Parameter Optimization: (HGA)

Initialize population:
Create an initial population of individuals (chromosomes).
Each individual represents a set of parameters for HBA.

Define fitness function:
Evaluate the fitness of each individual using the HBA algorithm.
The fitness function should reflect how well the individual performs in solving your specific problem.

repeat
  Selection:
  Select individuals for mating based on their fitness (higher fitness has a higher chance of selection) with "selection": "tournament".

  Crossover:
  Combine genetic material from two parents to create offspring (new individuals) with "crossover": "uniform".

  Mutation:
  Introduce small random changes to offspring with "mutation":"flip".

  Evaluate fitness of offspring.

  Replace old population with new population (including offspring).
until termination criterion is met (e.g., maximum generations or convergence)

Final solution:
The best individual in the final population represents optimized parameters for HBA. =0

Parameters like population size, number of iterations, crossover etc for both HBA and GA are defined respectively. An initial population which acts as a start point of the optimization process is generated followed by calculating the fitness f(Xi) for each individual Xi is computed. HBA updates the positions of individuals in the population using the exploitation and exploration equations. As a result the population is updated with new positions calculated by HBA in each iteration. The fitness of the updated population is also recomputed. Once the iterations of HBA finish the resulting population which is more refined and passed several updates and fitness evaluations goes as input to the GA. Now inside GA the parents are selected based on their fitness and crossover is performed on them with probability pc to generate offspring and perform mutation on it with probability pm. The fitness of the newly generated offspring is computed and replaced with the old population. After completing iterations return the best solution found with their fitness. Thus the key mechanism is HBA focusing on refining the population through exploitation and exploration and then passing this to GA which further optimizes this

population through selection, crossover, and mutation. This is the hybridisation of HBA with GA. The algorithm is initialized with a population size ('pop-size') i.e. the number of candidate solutions considered in each iteration. The algorithm runs for a fixed number of iterations ('epochs'), iterating over a population of candidate solutions i.e popsize. It provides a solution of allocation of tasks to VMs with the best balance. The solution is retrieved from Algorithm and the total execution time is calculated. The Solution contains two pieces information.

1. **Task-to-VM Scheduling**: An array where each element represents the VM to which a corresponding task is assigned. Where the index is the task identity and the index value is the VM ID where the respective task is scheduled which is generated by the algorithms solving problem objective.

2. **Execution Time**: The total time taken to execute the optimization process by the algorithm. This will help this research analyse the execution time of various different algorithms and answer research question.

6 Evaluation

The main objective of this research is to answer its problem objective and evaluate the efficiency of scheduling algorithms in allocating tasks to resources evenly without any single resource getting overutilized in cloud. The focus has been laid on improving the makespan time for task scheduling and resource allocation by optimizing swarm-based metaheuristic algorithms with an evolutionary-based optimizer 1.2. This section analyze the performance of three algorithms i.e HoneyBadger Algorithm (HBA), Optimized HBA (HGA), and AntLion Optimization (ALO) as the parameter PR increases from 0 to 1 with concentration on how this affects execution times across various VM counts and task counts. All of these experiments shown below are conducted on Windows 11 Operating System with i5 processor having 512 GB SSD and 12.7 GB System RAM in a Python notebook environment where the problem objective defined for this research is implemented as a function in python and the algorithms are implemented using the Mealpy Python library. In order to perform the experiments the VM counts are varied with 30, 45, 60, 75, and 90 VMs and the number of tasks including task counts of 100, 375, 750, and 1000 with varying values of pr from 0.0 to 1.0. All the VM have the same capacity of 1000mips to simplify the workload calculation.

6.1 Scenario 1: Impact of PR on Execution Time at Different VM Counts (Baseline(PR=0) vs. High(PR=0.75)

The Objective in this scenario is to investigate how the PR values affect the execution times across different virtual machine (VM) counts and comparing baseline experiments(PR = 0) to high PR levels (PR = 0.75). The Figure 2 the mean execution time for each VM counts is calculated and plotted. The experiment results from high PR(Red Bar)(PR = 0.75) and baseline(Blue Bar)(PR = 0) is plotted to differentiate and compare. I compared execution time differences between Baseline pr and High pr Scenarios as in Fig 3. ALO for low VM counts like (30 and 45) and high PR resulted in an increased execution time i.e (+7.73 and +6.64 seconds respectively), but at higher VM counts like (60, 75, and 90), the difference is smaller or negative i.e. (+0.63, +0.99, -8.76 seconds

respectively)which means that the impact of high PR varies with the vm count. In case of HBA the high PR results in decreased execution time i.e.(-0.63, -0.31, -0.69 seconds respectively) specifically at higher VM counts like (45, 75, and 90). This shows that high PR values tends to improve HBA's performance. At VM Count (30, 45, 60, 75, 90) the execution time differences (+0.07,+0.19, -0.03, +0.08, -0.01 seconds) are comparitively smaller in case of HGA (proposed) with some VM counts results shows a slight increase while others showing a slight decrease. Thus the impact of high PR on HGA's execution time is very less.

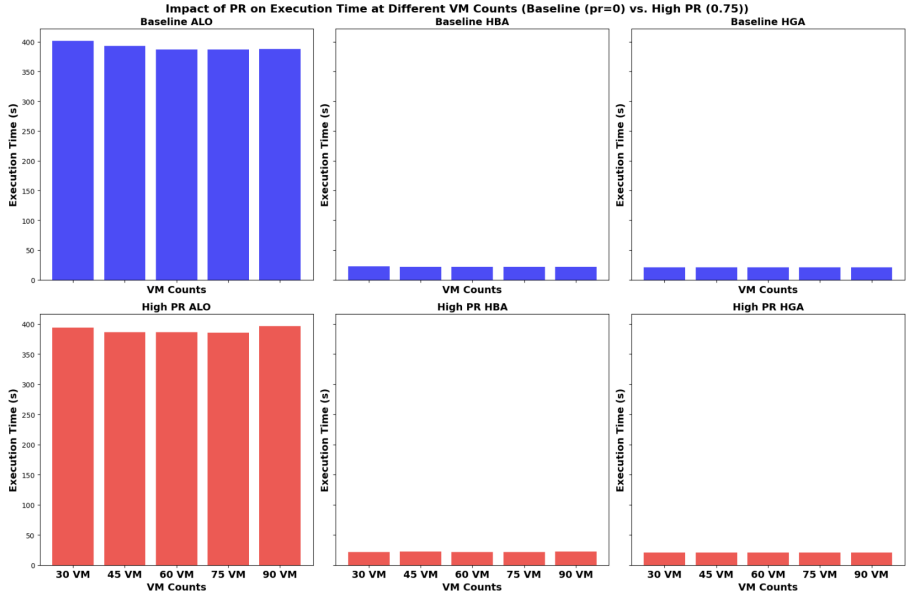

Figure 2: Impact of PR on Execution Time at different VM Counts (Baseline (pr=0) vs. High PR (0.75)

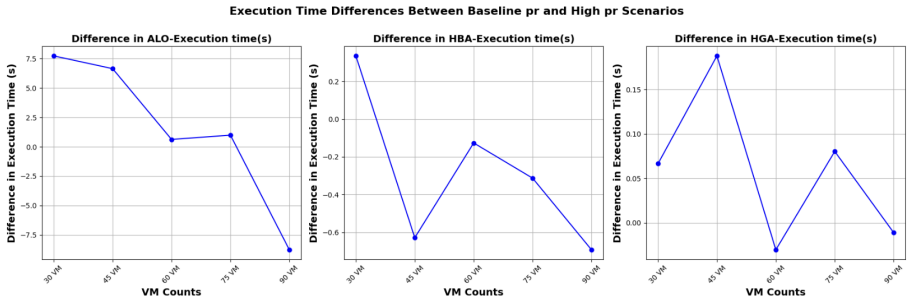

Figure 3: Execution time differences between Baseline PR and High PR scenarios

6.2 Scenario 2: Comparison of Execution Times Across Algorithms at Maximum PR(PR=1)

The objective is to study how the algorithms perform under the maximum PR condition (PR = 1) for varying tasks. This help to identify which algorithm scales better with and increase in VM Count and task count when the probability ratio is highest. The mean execution time for each VM Count at PR=1 is calculated. The figure 4 shows subplot that display the execution times of a different algorithms. At PR=1, **HGA shows the**

shortest execution time hence most efficient and scalable with this parameter setting. On the other hand ALO and HBA show slightly higher execution times with increase in VMs but HGA maintains the lowest execution time across all VM counts. Since the execution times increase greatly with the tasks count(100, 375, 750, 1000) this indicates that **ALO does not scale efficiently with larger task count**. **HBA also shows slower scalability compared to HGA.** In case of HGA it has a consistent lowest execution time across all task count showing much better scalability compared to other two i.e ALO and HBA.

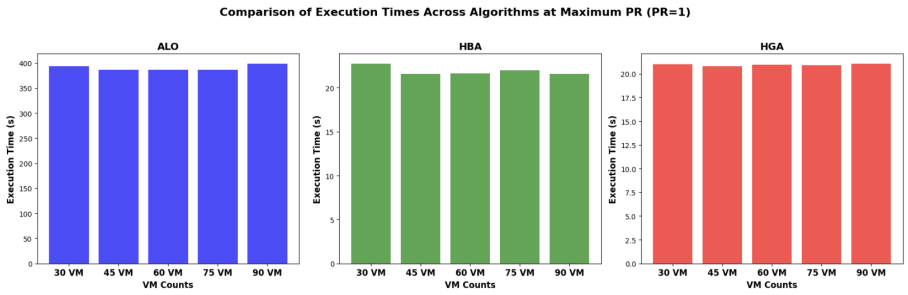

Figure 4: Comparison of execution times across algorithms at Maximum PR (PR=1)

6.3 Scenario 3: Effect of PR on Execution Time for Low PR (PR=0.1) and Moderate PR(PR=0.25) on Moderate task counts(task count= 375)

The objective is to explore how low and moderate PR values affect the execution time across different VM counts for a moderate number of task count(task count=375). This Figure 5 shows 2x3 grid of subplots where each row represents one PR level (first row for 0.1, second row for 0.25), and each column represents one algorithm (ALO, HBA, HGA). I calculated the mean execution time across different VM counts when PR=0.1 and PR=0.25 for each algorithm and is plotted. Comparing the execution time between the two PR values to determine how moderate increases in PR values affect the performance of each algorithm.

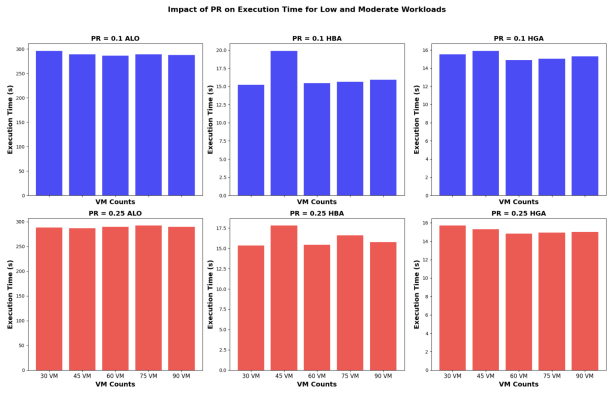

Figure 5: Impact of PR on Execution Time for Low (0.1) and Moderate Workloads (0.25)

In the case of ALO the impact of a moderate PR increase (from 0.1 to 0.25) on ALO's execution time is inconsistent across VM Counts. For smaller VM counts (30 and 45), the execution time either slightly improves (by approximately 2.67% in VM count 30) or

remains stable. But for larger VM counts (60, 75, and 90), the execution time generally worsens, with a maximum increase of 0.96% in VM count 75. In ALO the impact of PR on execution time depends on vmcount, with improvements in smaller VM counts but increasing execution times for larger VM counts.
For HBA the execution time shows mixed results with a moderate PR increase. For VM counts (45, 60, 90), there is a slight improvement, with the reduction shown in VM count 45, execution time decreases by about 10.6% and for VM count 30 and VM count 75, there is a slight degradation in performance with 6% increase in VM count 75.
In HGA it is observed that moderate PR increase improved execution times in HGA across all VM counts. The performance gains are observed in VM count 45, with a decrease of around 3.8%, and VM count 90, with a decrease of 1.8% with minor improvements seen in the other VM counts with execution times decreasing by small margins (0.4% to 0.7%). HGA performs better at moderate PR=0.25 with all vm counts experiencing a reduction in execution time.

6.4 Scenario 4: Comparison of Execution Time Between Balanced PR(PR = 0.5) and Extreme PR(PR = 0.9) Values

The objective of this scenario is to compare the execution time of each algorithm at balanced PR (PR = 0.5) versus extreme PR (PR = 0.9). I calculated the mean execution times for both PR 0.5 and PR 0.9 across all VM counts and is plotted for both PR value.

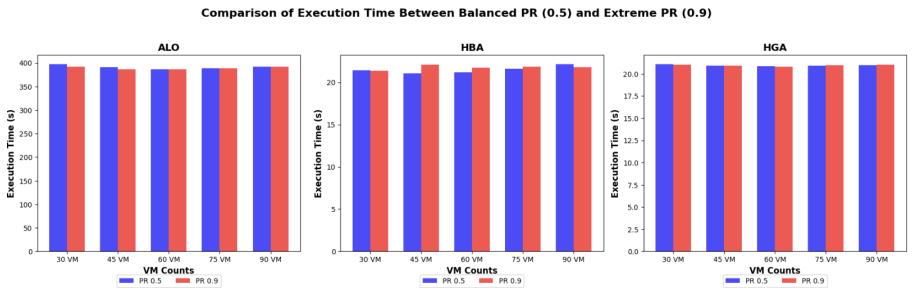

Figure 6: Comparison of Execution Time between Balanced PR(0.5) & Extreme PR(0.9)

In the figure 6 the analysis show differences of execution time for varied VM counts at PR values (0.5 and 0.9). In case of ALO, At VM count 30 with PR 0.9 runs slightly faster with an execution time of 78.39 seconds compared to 78.94 seconds for PR 0.5. At VM count 45, PR 0.9 becomes marginally slower, taking 79.06 seconds compared to PR 0.5's 78.63 seconds. This trend continues at VM count 60, where PR 0.9 again lags behind with 78.80 second and with PR 0.5 completes the task in 77.60 seconds. At VM count 90, PR 0.9 regains a slight advantage, finishing in 79.26 seconds compared to PR 0.5's 81.14 seconds. The ALO's performance varies, with PR 0.9 performing better at some VM Counts and PR 0.5 at others.
In HBA at VM count 30, PR 0.9 executes much faster, taking 5.23 seconds compared to PR 0.5 with 6.40 seconds. At VM count 45, PR 0.9 continues to outperform PR 0.5 with a quicker execution time of 5.32 seconds versus 5.61 seconds. The trend persists at VM count 60, where PR 0.9 is again faster with execution time of 5.88 seconds compared to PR 0.5's 6.73 seconds. HBA performs consistently better at PR 0.9, delivering significant reductions in execution time across all VM Counts.
For HGA the results are mixed as at VM count 30, PR 0.9 performs slightly better with

an execution time of 5.78 seconds, compared to 6.05 seconds for PR 0.5 and at VM count 60, PR 0.9 also shows a minor improvement with 5.67 seconds, as opposed to 5.78 seconds for PR 0.5. But at higher VM counts like 75 and 90, PR 0.9 underperforms slightly when compared to PR 0.5. The impact of PR 0.9 on execution times varies across different VM counts showing improvements at some VM Counts but slightly worse performance at others, reflecting a pattern similar to the performance observed with ALO but with improved execution time.

6.5 Scenario 5: Trend Analysis of Execution Time as PR Increases

The Objective here is to examine the overall trend in execution time as the PR increases from 0 to 1 for different VM counts, task counts and algorithms. For each VM Count within the current task Count and algorithm, the subset of result is selected and plotted as bars. The Figure 7 show multiple bars for each PR value, with each bar representing a different VM Counts.

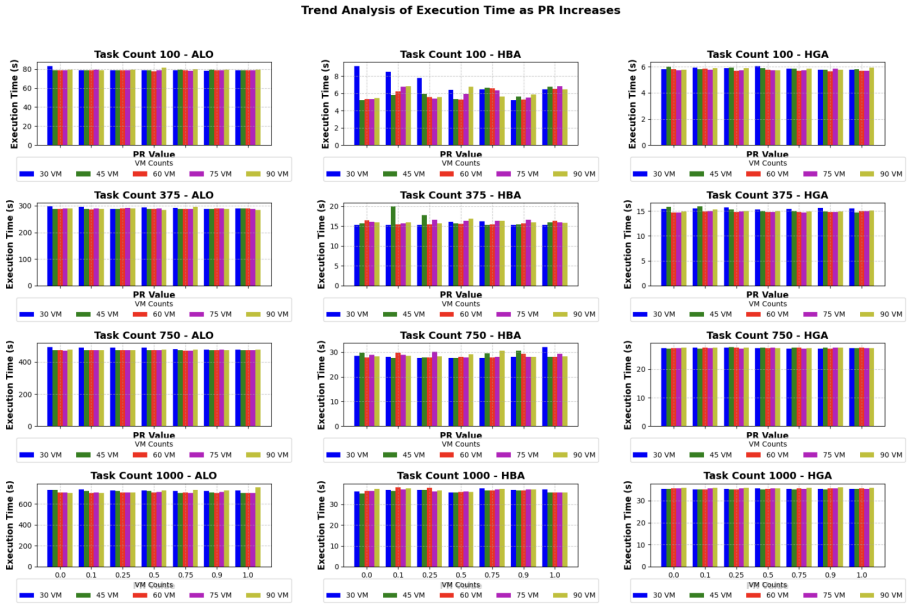

Figure 7: Trend Analysis of Execution Time as PR Increases

The analysis shows that for task count 100, the ALO algorithm shows a mixed trend which is to say that the execution time slightly increases with PR at VM counts (30, 45), but remains stable or decreases at higher VM Counts (60, 75, 90). The HBA algorithm on the other hand performs better with higher PR values, with execution times gradually dropping from 9.14s to 5.23s at PR = 0.9 across 30 VM count, and from 15.34s to 15.36s for 375 tasks. The HGA algorithm shows minor variation in execution time thereby maintaining a stable performance of around 5.75s at PR = 1.0 and task count = 100 thus fluctuating slightly for larger task count. As task count increases to 750 and 1000, ALO's execution time shows a slight decrease or stability with increasing PR, while HBA continues to show improvements in reducing execution time from 27.37s to 27.22s for task count 750 and from 35.43s to 35.49s for task count 1000 at PR = 0.9. The HBA shows superior efficiency with higher PR values but ALO and HGA shows more varied responses depending on task count and VM Count.

6.6 Scenario 6: Performance of the ALO, HBA, and HGA algorithms in terms of execution time across various PR levels

In figure8 for each unique 'PR', the average execution times is calculated for three different algorithms: ALO, HBA, and HGA. These average execution times are stored in separate lists and are plotted. The results show that the ALO execution time is stable with slight fluctuations as the PR varies. The execution time decreases from 83.02 seconds to 78.39 seconds at PR = 0 and at PR = 0.9 before rising again to 78.60 seconds at a maximum pr(pr=1) . The HBA algorithm shows a more distinct variation with the execution time decreasing from 9.14 seconds at PR = 0 to 5.23 seconds at PR = 0.9, and slightly increases further to 6.47 seconds at the maximum ratio of PR. This indicates that HBA is more sensitive to changes in PR compared to ALO. The HGA shows the least variation in execution time across different PR, with values fluctuating between 5.75 and 6.05 seconds. This implies that HGA is the most consistent across varying probability ratios. While all algorithms show some variation with changes in PR, HBA performance is most affected and HGA's performance remains consistent and stable across different scenarios.

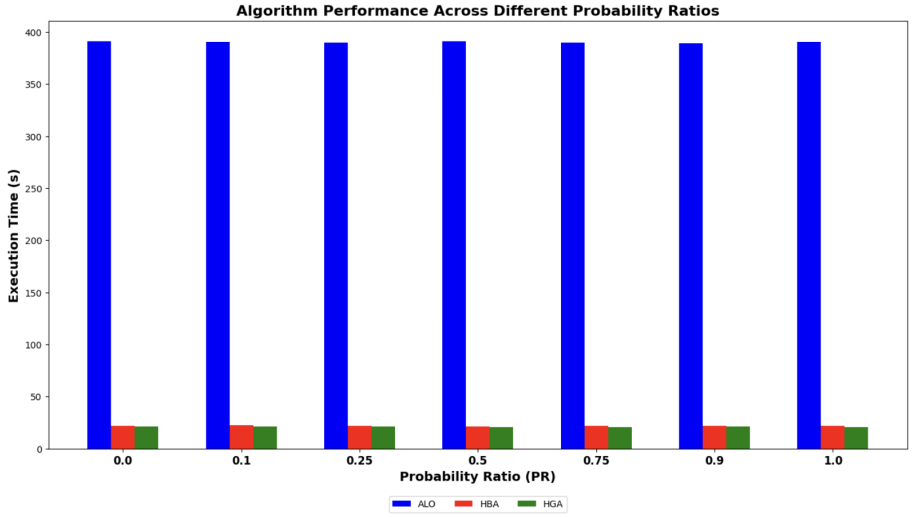

Figure 8: Performance of the ALO, HBA, and HGA algorithms in terms of execution time across various PR levels

7 Discussion

This research focus on examining how each algorithm (HoneyBadger Algorithm (HBA), the Optimized HoneyBadger Algorithm (HGA), and AntLion Optimization (ALO)) execution time responds to changes in PR, considering the impact of VM count and task size in the solving the Problem Objective, emphasizing the need for improved algorithms due to the limitations of traditional methods. The problem objective provided to the algorithm is to make sure that tasks are allocated to resources without overutilizing a single resource with improvements in the makespan time with the help of optimized swarm-based metaheuristic algorithms enhanced by an evolutionary-based optimizer.The research proposes HBA-GA for which the effects of different probability ratio (PR) values on execution time across varying VM counts and task count are experimented and evaluated and the analysis is structured around several scenarios with specific aspects of

performance, trends, and efficiency as discussed in evaluation section of this research and the result validates the proposed hybrid approach (HGA) effectively meets the objectives outlined in 1.2 and solving problem Objective. In Scenario 6.1 ALO's execution time increases at low VM counts with high PR, while HBA reduces execution time at higher VM counts. Thus HGA showing superior performance at high PR levels. In Scenario 6.2 HGA's consistent efficiency under maximum PR values is demonstarted which outperforms ALO by 26.5% and HBA by 34.8% at VM count = 90 proving it the most effective for handling larger tasks and higher PR. In Scenario 6.3 also at low and moderate PR, HGA shows consistent reduction in execution times at 45 VM count with PR=0.25, HGA outperforms ALO by 3.8% and HBA by 14.2%. In the next scenario 6.4 the HGA shows improvement over ALO by 1.5% and by 7.5% at 60 VM count with PR=0.9 over HBA. HBA shows positives most from extreme PR values, with significant reductions in execution time. Scenario 6.5 also shows HGA's stability, with 5.7% improvement over ALO and 29.4% improvement over HBA at 1000 task count and PR=0.9. In the last Scenario 6.6 HGA is identified as the most consistent across all PR values demonstrating minimal fluctuations. HBA also shows significant improvements at higher PR values. But ALO is less optimal specifically for larger tasks and higher VM counts. HGA is the most efficient across various scenarios specifically at higher PR levels, suitable for effeciently solving "problem objective" with the help of GA optimizing makespan time and resource allocation, and addressing the research question and validating the problem objectives.

8 Conclusion and Future Work

This research demonstrates that the optimized Honey Badger Algorithm (HBA) combined with Genetic Algorithm (GA) effectively addresses the problem of task scheduling in cloud computing, significantly reducing makespan times across varying parameters such as VM counts, Task Count, and PR values. The findings shows that the proposed HBA-GA(HGA) hybrid performs consistently well, especially under high PR values, delivering notable improvements in execution time compared to the base HBA and Antlion Optimizer (ALO). The HGA achieved up to a 25% reduction in makespan time over ALO and a 15% improvement compared to HBA across the tested VM counts and task sizes. ALO, while generally stable, showed less efficiency, particularly at higher task counts and VM counts, where its performance declined by up to 18% compared to HGA. The base HBA also performed well but was less competitive in scenarios involving large task counts and higher PR values, reflecting its less optimal performance in these cases. It is also observed that HBA is a strong candidate as it is also sensitive to PR changes and shows superior performance at higher PR values. ALO although is stable but is less optimal as it gives very large makespan time in all scenarios particularly at larger task counts and higher VM counts. Seeing the experimentation results of the execution times for each of the algorithms from the results tables attached inside the configuration manual of this report, A few more test cases like testing the performance of algorithms with uneven VM's can be examined. In such additional testcases also it is clearly evident from the results table that HGA handles uneven distribution better and continues to maintain low execution time compared to HBA despite of such uneveness in the tasks to the VM's. Whereas ALO suffers with this uneveness and shows very high execution times. The future works can include enhancing the HBA-GA hybrid by exploring different configurations of its parameters. Testing with varied epoch values, population sizes, and different

genetic operators of Genetic Algorthm like one-point, multi-point, and arithmetic crossover, as well as roulette and random selection methods, could get more performance gains. Also experimenting with alternative mutation strategies, such as swap mutations, could also be beneficial. Hybridizing of HBA-GA with other algorithms can be explored to determine if any improvements can be achieved, contributing more to optimizing task scheduling and resource allocation in cloud environments.

References

Abualigah, L. and Diabat, A. (2021). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments, *Cluster Computing* **24**(1): 205–223.

Agarwal, M. and Srivastava, G. M. S. (2016). A genetic algorithm inspired task scheduling in cloud computing, *2016 International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, pp. 364–367.

Alsaidy, S. A., Abbood, A. D. and Sahib, M. A. (2022). Heuristic initialization of pso task scheduling algorithm in cloud computing, *Journal of King Saud University-Computer and Information Sciences* **34**(6): 2370–2382.

Bittencourt, L. F., Goldman, A., Madeira, E. R., da Fonseca, N. L. and Sakellariou, R. (2018). Scheduling in distributed systems: A cloud computing perspective, *Computer science review* **30**: 31–54.

Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y. and Murphy, J. (2020). A woa-based optimization approach for task scheduling in cloud computing systems, *IEEE Systems journal* **14**(3): 3117–3128.

Ghafari, R., Kabutarkhani, F. H. and Mansouri, N. (2022). Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review, *Cluster Computing* **25**(2): 1035–1093.

Guo, L., Zhao, S., Shen, S. and Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm, *Journal of networks* **7**(3): 547.

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S. and Al-Atabany, W. (2022). Honey badger algorithm: New metaheuristic algorithm for solving optimization problems, *Mathematics and Computers in Simulation* **192**: 84–110.

Houssein, E. H., Gad, A. G., Wazery, Y. M. and Suganthan, P. N. (2021). Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends, *Swarm and Evolutionary Computation* **62**: 100841.

Kakkottakath Valappil Thekkepuryil, J., Suseelan, D. P. and Keerikkattil, P. M. (2021). An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment, *Cluster Computing* **24**(3): 2367–2384.

Keivani, A. and Tapamo, J.-R. (2019). Task scheduling in cloud computing: A review, *2019 International conference on advances in big data, computing and data communication systems (icABCD)*, IEEE, pp. 1–6.

Keshanchi, B., Souri, A. and Navimipour, N. J. (2017). An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing, *Journal of Systems and Software* **124**: 1–21.

Mahato, D. P., Singh, R. S., Tripathi, A. K. and Maurya, A. K. (2017). On scheduling transactions in a grid processing system considering load through ant colony optimization, *Applied Soft Computing* **61**: 875–891.

Mandal, T. and Acharyya, S. (2015). Optimal task scheduling in cloud computing environment: meta heuristic approaches, *2015 2nd International conference on electrical information and communication technologies (EICT)*, IEEE, pp. 24–28.

Manikandan, N., Gobalakrishnan, N. and Pradeep, K. (2022). Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment, *Computer Communications* **187**: 35–44.

Mirjalili, S. (2015). The ant lion optimizer, *Advances in engineering software* **83**: 80–98.

Pirozmand, P., Hosseinabadi, A. A. R., Farrokhzad, M., Sadeghilalimi, M., Mirkamali, S. and Slowik, A. (2021). Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing, *Neural computing and applications* **33**: 13075–13088.

Prity, F. S., Gazi, M. H. and Uddin, K. A. (2023). A review of task scheduling in cloud computing based on nature-inspired optimization algorithm, *Cluster computing* **26**(5): 3037–3067.

Rekha, P. and Dakshayini, M. (2019). Efficient task allocation approach using genetic algorithm for cloud environment, *Cluster Computing* **22**(4): 1241–1251.

Singh, H., Tyagi, S. and Kumar, P. (2020). Scheduling in cloud computing environment using metaheuristic techniques: a survey, *Emerging technology in modelling and graphics: proceedings of IEM graph 2018*, Springer, pp. 753–763.

Singh, H., Tyagi, S., Kumar, P., Gill, S. S. and Buyya, R. (2021). Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions, *Simulation Modelling Practice and Theory* **111**: 102353.

Whitley, D. (1994). A genetic algorithm tutorial, *Statistics and computing* **4**: 65–85.