A Project Report

Workflow Scheduling in Cloud using DGWO

Submitted in partial fulfillment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

V. Srilaxmi

1602-19-733-110

M. Srividya

1602-19-733-111

Under the guidance of

S. Vinay Kumar

Assistant Professor

Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2023

Vasavi College of Engineering(Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

DECLARATION BY THE CANDIDATE

I, **V. Srilaxmi,** bearing hall ticket number, **1602-19-733-110** and **M. Srividya** bearing hall ticket number **1602-19-733-111** hereby declares that the project report entitled **"Workflow Scheduling in Cloud using DGWO"** under the guidance of **S.Vinay Kumar,** Assistant Professor , Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering**.

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

V.Srilaxmi,
1602-19-733-110

M.Srividya,
1602-19-733-111

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"Workflow Scheduling in Cloud using DGWO"** is being submitted by **V. Srilaxmi** bearing **1602-19-733-110** and **M.Srividya** bearing **1602-19-733-111,** in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by them under my guidance.

S.Vinay Kumar ,
Assistant Professor,
Internal Guide

Dr. T. Adilakshmi,
Professor & HOD,
Dept. of CSE

ACKNOWLEDGEMENT

We would like to express my heartfelt gratitude to, our external project guide, for his valuable guidance and constant support, along with his capable instructions and persistent encouragement.

We are grateful to our Head of the Department, Dr. T. Adilakshmi, Principal, Dr. M. Sunitha Reddy, Internal guide S. Vinay Kumar and the management of the institute, for providing this opportunity.

V. Srilaxmi
1602-19-733-110

M. Srividya
1602-19-733-111

ABSTRACT

In cloud computing environments, virtualization technology is used to create Virtual Machines (VMs) which can host and execute workflows composed of dependent tasks. To optimize the utilization of available resources, it is important to schedule these workflows efficiently. Task scheduling involves assigning the dependent tasks of a flow of work to the available VMs in the cloud while trying to minimize the total completion time.

The problem of scheduling tasks in cloud computing is recognized as NP-hard, which means that finding an optimal solution can be computationally infeasible for large-scale problems. To overcome this challenge, optimization algorithms such as genetic algorithms, particle swarm optimization, and simulated annealing have been proposed in the literature. However, the performance of these algorithms may be compromised when addressing optimization problems with medium and high dimensions.

To tackle this difficulty, the scheduling procedure can be formulated as a dual-objective minimization problem, taking into account both computation and data transmission expenses. The computation cost represents the total execution time of the workflow, while the data transmission cost represents the total amount of data transferred between the VMs. To optimize this multi-objective problem, a variety of algorithms can be used, including the DGWO (Discrete Grey Wolf Optimizer) algorithm.

The DGWO algorithm, inspired by the social behavior of grey wolves, is a modern metaheuristic optimization algorithm. This algorithm has been shown to perform well in a variety of optimization problems, including job scheduling, task allocation, and feature selection. By using the DGWO algorithm to solve the workflow scheduling problem, it is possible to achieve better performance than traditional optimization algorithms, particularly for medium and high-dimensional problems.

TABLE OF CONTENTS

Abstract……..…….(v)
List of figures…….……...(vii)
1. Introduction.....….08
    1.1. Overview.....08
    1.2. Motivation…….……...09
    1.3. Problem Definition.....09
    1.4. Objectives…….……11
    1.5. Scope……....…….…….……11
2. Literature Survey..…….…….……13
3. Proposed Methodology…….……..…….……...17
    3.1. Software Requirements…….……..…….…17
    3.2. Hardware Requirements…….……..……...…….....19
    3.3. Algorithm and Techniques…….……..……...……..19
4. Experimental Results……..……..…….……34
5. Testing……..…….……..…….……...40
6. Conclusion……42
7. Future Work…….……43
    References……45

LIST OF FIGURES

Fig. 1.1 – Task to VM Mapping…… 10

Fig. 3.1 – Hierarchy of Grey Wolves……...20

Fig. 4.1 – Execution costs incurred using GWO and DGWO for input1……34

Fig. 4.2 – Execution costs incurred using GWO and DGWO for input2……35

Fig. 4.3 – Execution costs incurred using GWO and DGWO for input3……36

Fig. 4.4 – Execution costs incurred using GWO and DGWO for input4……37

Fig. 4.5 – Execution costs incurred using GWO and DGWO for input5……38

CHAPTER-1
INTRODUCTION

1.1 Overview

In cloud computing, scheduling refers to the process of determining the order and timing of tasks or jobs to be executed by a computer system. It is a critical aspect of resource management in computing systems, as efficient scheduling can lead to improved system performance, resource utilization, and overall throughput.

Workflow scheduling is a specific type of scheduling that is used in cloud computing environments to manage the execution of complex workflows. A workflow is a sequence of tasks that must be executed in a specific order to achieve a desired outcome. In cloud computing, workflows are often used to model complex business processes or scientific experiments that require the coordination of multiple resources and services.

Workflow scheduling involves the allocation of resources, such as virtual machines, storage, and network bandwidth, to the various tasks within a workflow. The goal is to ensure that the workflow is executed within the given constraints, such as deadline and budget, while also optimizing resource utilization and minimizing the cost of execution.

There are various scheduling algorithms that can be used for workflow scheduling, such as heuristics, metaheuristics, and optimization algorithms. The selection of the algorithm is contingent upon the unique demands of the workflow and the attributes of the computing environment.

Workflow scheduling is a critical aspect of cloud computing, as it enables the efficient execution of complex workflows that are essential for many business and scientific applications. In this report, we will explore the state-of-the-art workflow scheduling schemes in cloud computing and provide a subjective understanding of their strengths and weaknesses.

1.2 Motivation

The main motivation behind workflow scheduling is to ensure that complex workflows can be executed efficiently in a cloud computing environment. This involves the allocation of resources to the various tasks within the workflow to ensure that they are executed within the given constraints, such as deadline and budget, while optimizing resource utilization and minimizing the cost of execution.

Workflow scheduling is essential for many business and scientific applications that require the coordination of multiple resources and services to achieve a desired outcome. For example, in a business setting, workflows may be used to automate the processing of customer orders or to manage the distribution of goods and services. In a scientific setting, workflows may be used to model complex simulations or experiments that require the coordination of multiple resources, such as computing power and storage.

Efficient workflow scheduling can lead to significant improvements in system performance, resource utilization, and overall throughput, which can have a direct impact on business productivity and scientific discovery. Therefore, the main motivation behind workflow scheduling is to ensure that complex workflows can be executed effectively and efficiently in cloud computing environments, enabling businesses and researchers to achieve their goals with greater speed and accuracy.

1.3 Problem Definition

Workflow scheduling is a critical problem in cloud computing, which involves the efficient allocation of resources to execute complex workflows while meeting quality of service (QoS) requirements. To solve this problem, conventional optimization techniques like genetic algorithms and particle swarm optimization (PSO) have been employed. However, these techniques often suffer from issues such as premature convergence and slow convergence rates.

In this project, we suggest employing the Distributed Grey Wolf Optimizer (DGWO) for solving the workflow scheduling problem. DGWO is an emerging optimization algorithm that draws inspiration from the hunting behavior of grey wolves. It has demonstrated superior performance compared to

conventional optimization techniques in diverse applications, including scheduling and resource allocation problems.

Our objective is to demonstrate the effectiveness of DGWO in solving the workflow scheduling problem. We consider a scenario where a set of tasks with precedence constraints need to be scheduled on a set of heterogeneous virtual machines. The problem is formulated as a multi-objective optimization problem that aims to minimize cost of execution.

We compare the performance of DGWO with that of other state-of-the-art algorithms, namely, Grey Wolf Optimizer (GWO) in terms of solution Our experimental results show that DGWO outperforms GWO in terms of solution quality indicating its potential as a promising optimization technique for workflow scheduling in cloud computing.

The primary problem addressed in this project is Scheduling workflows within cloud computing environments using the Discrete Grey Wolf Optimizer (DGWO) algorithm. Specifically, this project aims to investigate the effectiveness of the DGWO algorithm for minimizing computation and data transmission costs associated with workflow scheduling. This project will compare the performance of the DGWO algorithm against traditional optimization algorithms and evaluate its scalability and robustness in handling large-scale workflow scheduling problems.

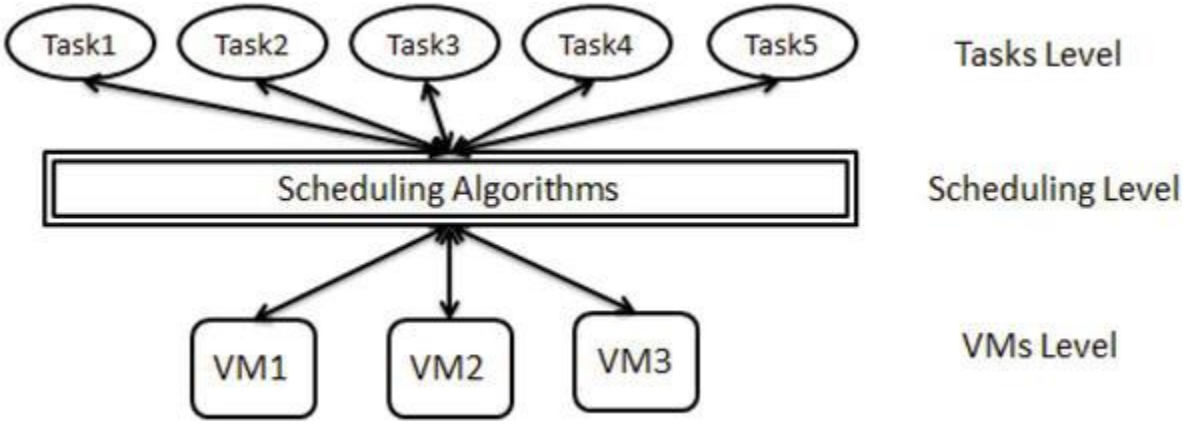


Fig 1.1 Task to VM mapping

1.4 Objectives

The objective of this study is to investigate the effectiveness of the Distributed Grey Wolf Optimizer algorithm for optimizing Scheduling workflows within cloud computing environments.
Specifically, the study aims to achieve the following objectives:

1. To develop a mathematical model for Scheduling workflows within cloud computing environments using the DGWO algorithm, taking into consideration various constraints such as task dependencies, resource availability, and deadline requirements.

2. To calculate the efficiency of the DGWO algorithm in terms resource utilization, which is the percentage of available resources used in the scheduling process.

3. To analyze the scalability and robustness of the DGWO algorithm in handling large-scale workflow scheduling problems and identify its limitations and areas for improvement.

4. To provide insights into the potential of the DGWO algorithm for improving Scheduling workflows within cloud computing environments and identify potential applications and future research

1.5 SCOPE

The scope for the DGWO algorithm for workflow scheduling in the cloud is broad, and the algorithm has the potential to be applied to various real-world scenarios. Some of the areas where the algorithm can be applied include:

1. Cloud-based data analytics: With the increasing amount of data generated by various industries, there is a growing need for efficient data analytics systems. The application of the DGWO algorithm in optimizing the scheduling of data analytics workflows in the cloud can enhance system performance and efficiency.

2. Machine learning: Machine learning algorithms require significant computing resources to train and optimize models. The DGWO algorithm is used to optimize the scheduling of machine learning

workflows in the cloud, improving the speed and efficiency of the training process.

3. Bioinformatics: Bioinformatics involves the analysis of biological data using computational tools. By leveraging the DGWO algorithm, it is possible to optimize the scheduling of bioinformatics workflows in the cloud, leading to improved analysis efficiency and accuracy.

4. Internet of Things : IoT to create large volumes of data that require efficient processing and analysis. The DGWO algorithm can be applied to maximize the scheduling of IoT workflows in the cloud, improving performance and efficiency of the system.

5. Cloud gaming: Cloud gaming requires high-performance computing resources to provide seamless gaming experiences to users. The DGWO algorithm is applied to maximize the scheduling of cloud gaming workflows, improving the speed and responsiveness of the system.

Overall, the scope for the DGWO algorithm for workflows in the cloud is to wide-ranging and the algorithm has the potential to significantly improve the performance and efficiency of various cloud-based applications.

CHAPTER 2

LITERATURE SURVEY

Here is a literature survey for the DGWO algorithm for scheduling workflows in the cloud:
Y.Peng et.al[1] proposes a novel optimization algorithm, called Dynamic Grouping Grey Wolf Optimization Algorithm (DGGA), for solving the workflow scheduling problem in cloud computing environments.

The DGGA algorithm, proposed in this study, is derived from the Grey Wolf Optimization (GWO) algorithm, a novel optimization technique inspired by the hunting behavior of grey wolves.. The authors introduce a dynamic grouping strategy in the DGGA algorithm to improve its performance in solving the workflow scheduling problem. The dynamic grouping strategy involves dividing the tasks in the workflow into multiple groups based on their characteristics, such as processing time and resource requirements. The groups are then optimized separately using the GWO algorithm, and the solutions are combined to form a final solution.

The performance of the proposed DGGA algorithm is assessed by the authors through experimentation using a collection of benchmark workflows of varying sizes and characteristics. The results obtained from DGGA are compared against two well-established optimization algorithms, namely Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). Evaluation metrics such as makespan, cost, and success rate are employed to measure and compare the performance of the algorithms.

The experimental findings demonstrate that the proposed DGGA algorithm surpasses both PSO and GA in terms of makespan and cost, highlighting its potential as a highly effective optimization technique for scheduling workflows in cloud computing environments. Additionally, the authors perform a sensitivity analysis to assess the algorithm's robustness when confronted with variations in input parameters.

In summary, the research article presents a notable contribution to the field of workflow scheduling in cloud computing by introducing a novel optimization algorithm that enhances the performance of current techniques. The introduction of the dynamic grouping strategy in DGGA offers a promising

approach for effectively addressing intricate workflow scheduling problems that involve multiple objectives and constraints.

R.Ma et al. [2] presents an improved Grey Wolf Optimization (GWO) algorithm for solving the workflow of scheduling problem in cloud computing environments.

The enhanced GWO algorithm proposed in this study integrates a novel heuristic operator known as "task swapping" to bolster the exploration capability of the algorithm. The task swapping operator involves swapping the positions of two tasks in the workflow to generate new solutions. The authors also introduce a new fitness function that considers both the make span and the cost of execution, which are two important objectives in the workflow scheduling problem.

The authors conduct an evaluation of the performance of the proposed improved GWO algorithm by conducting experiments on a set of benchmark workflows of varying sizes and characteristics. They compare the results obtained from the improved GWO algorithm with three state-of-the-art optimization algorithms: Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and an earlier version of the GWO algorithm. The evaluation metrics encompass make span, cost, and convergence speed, which are used to assess and compare the performance of the algorithms.

Overall, the research article makes a significant contribution to work of workflow scheduling in the cloud computing by introducing a new optimization algorithm that improves the performance of existing techniques. The task swapping operator and the new fitness function are promising approaches for solving complex workflow scheduling problems with multiple objectives and constraints. The experimental results obtained from the study exhibit the efficacy of the proposed algorithm, thereby offering valuable insights for future research endeavors in the field.

S. Venkatesan et al.[3] proposes a hybrid optimization algorithm based on the Genetic Algorithm (GA) and Grey Wolf Optimization (GWO) techniques for solving the workflow scheduling problem in cloud computing environments. The goal of the workflow scheduling problem is to give a task available resources in such a way maximizes the performance of the workflow while satisfying various constraints such as deadline and budget.

The hybrid algorithm proposed in this study combines the strengths of both Genetic Algorithm (GA) and Grey Wolf Optimization (GWO). GA is a widely recognized optimization algorithm that employs genetic operators like selection, crossover, and mutation to generate new solutions. On the other hand, GWO is a relatively novel optimization algorithm that draws inspiration from the hunting behavior of grey wolves. GWO employs four fundamental operators, namely initialization, updating, exploration, and exploitation, to explore and exploit the search space for optimal solutions.

Proposed hybrid algorithm involves three stages. In the first stage, the initial population is generated using the GWO algorithm. In the second stage, the GA operator is used to create new solutions by selecting, crossing over, and mutating the best solutions generated in the first stage. In the third stage, the GWO operator is used to refine the solutions obtained from the second stage to get the accurate \final answer.

The authors assess the performance of the proposed algorithm by conducting experiments on a diverse set of benchmark workflows that vary in terms of size and characteristics.They compare the results of the proposed hybrid algorithm with three other state-of-the-art optimization algorithms, namely, PSO, Ant Colony Optimization (ACO), and GWO. The evaluation metrics include makespan, cost, and energy consumption.

The experimental results indicate that the proposed hybrid algorithm surpasses the other algorithms in terms of makespan, cost, and energy consumption. This suggests that the hybrid algorithm holds significant promise as an effective optimization technique for scheduling workflows in cloud computing environments.

Overall, the research article represents a novel hybrid optimization algorithm combines the advantages of GA and GWO for solving the workflow schedule problem in cloud computing environments. The experimental results demonstrate effectiveness of proposed algorithm and provide valuable information.

X. Peng et al. [4] proposes a optimized scheduling algorithm based on grey wolf optimization (GWO) and (DE) for cloud workflow applications. The algorithm uses the DE algorithm to generate initial solutions and then applies the GWO algorithm to refine the solutions and improve the optimization performance.

Y. Feng et al.[5] proposes a hybrid grey wolf optimization (H-GWO) algorithm is proposed for scheduling cloud-based workflow applications. The algorithm uses a strategy that combines the standard GWO algorithm with a modified version of the algorithm, which incorporates local search and elitism.

Overall, these research articles illustrate the effecienciness and versatility of the DGWO algorithm and the variations for scheduling workflows in the cloud. The use of hybrid and improved versions of the algorithm further enhances the optimization performance and convergence speed of the algorithm.

CHAPTER-3

PROPOSED METHODOLOGY

3.1 SOFTWARE REQUIREMENTS

1. Java

Java is an object-oriented, high-level programming language that emphasizes minimizing dependencies in implementation. It is widely recognized as the leading programming language for Internet of Things (IoT) and enterprise architecture.

Java is general purpose programming language that is a programmer writes code once and can run it anywhere. It is platform independent, compiled code of the java can run on any platform that supports java.

Java shares a similar syntax with C and C++, but it offers fewer low-level capabilities compared to both languages. Java code is compiled into bytecode, a platform-independent format that can be executed on any device with a Java Virtual Machine (JVM) installed. This unique feature enables Java programs to run seamlessly on various platforms, ranging from small embedded systems to extensive enterprise servers.

Java has a rich set of libraries and frameworks that provide developers with tools for building a wide different type of applications including many things. The language is known for its simplicity, readability, and ease of use, as well as its strong community support and rich ecosystem of third-party tools and libraries.

Java is also widely used in industry, with many large organizations relying on Java-based systems for their mission-critical applications. The language has a large and active developer community, which continues to contribute to its growth and evolution.

2. Visual Studio Code

Visual Studio is a comprehensive integrated development environment (IDE) that enables you to complete the entire development cycle in one place. From writing, editing, debugging, to building code, you can perform all these tasks within this IDE before deploying your project. In addition to code editing and debugging, Visual Studio offers a plethora of other features to enhance every step of the software development process, including compilers, code completion tools, source control, extensions, and much more.

Visual Studio offers a vast array of capabilities and language support, allowing you to progress from creating a simple "Hello World" program to building and deploying complex applications. With this IDE, you can create, test, and debug various applications, including .NET and C++ programs, edit ASP.NET pages in the web designer view, utilize .NET to develop cross-platform mobile and desktop applications, or build responsive Web UIs using C#.
Developed by Microsoft, Visual Studio is an integrated development environment (IDE) utilized for designing applications across Windows, Android, iOS, and web platforms.

Visual Studio offers an extensive suite of tools and functionalities to assist developers in writing, testing, and debugging their code. It comprises a code editor that features syntax highlighting, code completion, and code refactoring, as well as a graphical debugger that aids in detecting and resolving code issues. Additionally, the IDE comes equipped with built-in support for version control systems like Git and Subversion, and seamlessly integrates with various third-party tools and services.

Visual Studio supports multiple programming languages, including C#, C++, Python, Java, JavaScript, and more. It provides project templates and wizards to help developers get started quickly and easily, and it includes tools for building and packaging applications for deployment.

In addition to its core features, Visual Studio also offers a variety of extensions and add-ons that can be installed to enhance its capabilities. These include tools for mobile development, cloud development, game development, and more.

Overall, Visual Studio is a powerful and versatile IDE that is widely used by developers around the world for building a wide range of applications. Its rich set of features and tools make it a popular choice for both individual developers and large development teams.

3.2 HARDWARE REQUIREMENTS

The hardware requirements for running this model in visual studio are:

- A 64-bit operating system (Windows, macOS, or Linux).
- 4 GB of RAM (8 GB or more recommended).
- A multi-core processor (such as Intel i5 or i7).

3.3 ALGORITHM

Hierarchy of Grey Wolves

The DGWO algorithm is a metaheuristic optimization technique that takes inspiration from the social behavior of grey wolves. It mimics the hierarchical social structure observed in grey wolf packs, which comprises four distinct levels of wolves:

1.Alpha Wolves: These are the pack leaders, responsible for guiding and coordinating the group's actions. In the DGWO algorithm, the alpha wolf is represented by the best solution found thus far.

2. Beta Wolves: Beta wolves are second in command after the alpha wolf and are responsible for assisting the alpha wolf in its duties. In the DGWO algorithm, the beta wolves are represented by the second-best solution found so far.

3. Delta Wolves: The delta wolves are responsible for patrolling the territory of the pack and reporting

back to the alpha wolf about potential prey or threats. In the DGWO algorithm, the delta wolves are represented by the third-best solution found so far.

4. Omega Wolves: The omega wolves are the lowest ranking members of the pack and are responsible for following the commands of the wolves. In the DGWO of the omega wolves are represented by the rest of the solutions in the search space that are not among the alpha, beta, or delta wolves.

The DGWO algorithm uses the position of the alpa,beta,delta wolves to guide the search process towards the optimal solution. The position of each wolf is represented by a vector of variables that correspond to the decision variables of the optimization problem. At each iteration of the algorithm, the place of each wolf is updated based on a set of rules that mimic the social behavior of wolves.

Overall, the hierarchical social structure of grey wolves used in the GWO algorithm provides a powerful framework for conducting a metaheuristic search for the optimal solution of an optimization problem.

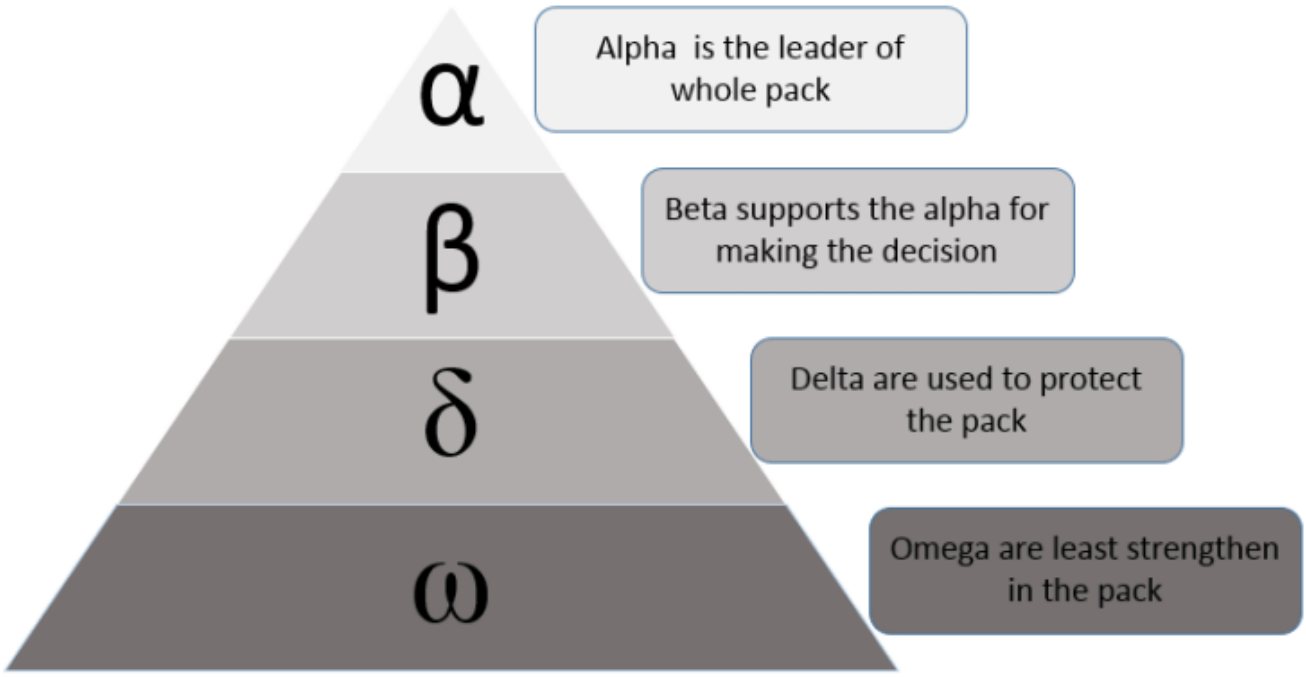


Fig 3.1 Hierarchy of Grey Wolves

3.3.1 Grey Wolf Optimizer (GWO)

Introduction

Inspired by the social behavior of grey wolves in nature, the Grey Wolf Optimizer (GWO) is a metaheuristic optimization algorithm that was introduced by Mirjalili et al. in 2014. It has gained widespread recognition for resolving diverse optimization problems in various domains. GWO is a population-based algorithm that emulates the hunting and leadership hierarchy of grey wolves to discover the best possible solution. This article delves into the intricacies of the GWO algorithm.

Initialization

The GWO algorithm begins by initializing a population of candidate solutions. The population size is typically predetermined, and each candidate solution represents a potential solution to the problem at hand. The initial positions of the candidate solutions are generated randomly within the defined search space. Additionally, three alpha, beta, and delta wolves are initialized to represent the three best solutions discovered up to that point.

Grey Wolf Hierarchy

The GWO algorithm simulates the social hierarchy and hunting behavior of grey wolves to optimize the solution. The grey wolves are divided into four levels of hierarchy, namely alpha, beta, delta, and omega. Alpha wolves are the most dominant and lead the hunting. Beta wolves follow alpha wolves, while delta and omega wolves are considered to be weak and follow the other wolves. In the GWO algorithm, the alpha, beta, and delta wolves symbolize the three best solutions obtained during the optimization process. These wolves represent the most promising solutions discovered thus far. On the other hand, the omega wolf represents a random solution within the search space.

Hunting Mechanism

The hunting mechanism of wolves involves three main steps: encircling prey, attacking prey, and searching for prey. In the GWO algorithm, these steps are represented by three mathematical operations,

namely encircling, attacking, and searching. In the encircling step, the positions of alpha, beta, and delta wolves are used to encircle the prey. During the attacking step of the algorithm, the wolves' positions are modified to approach the prey. On the other hand, during the searching step, the wolves' positions are updated to explore uncharted territories within the search space.

Updating Positions

The positions of the candidate solutions update using the positions of the alpha, beta, delta wolves. The update equations for each and every candidate soln are based on positions of the three wolves and a random number. The positions are updated using a weighted average of the positions of the three wolves. The update process persists until a predefined stopping criterion is fulfilled. This criterion can be a maximum number of iterations reached or the attainment of a satisfactory solution.

Conclusion

The Grey Wolf Optimizer algorithm is a powerful optimization technique that has been successfully applied to a variety of problems in different domains. The algorithm is designed to simulate the social behavior and hunting mechanism of grey wolves in nature. It adopts a hierarchical structure to emulate the leadership and hunting behavior observed in wolf packs.The algorithm is easy to implement and requires few parameters to be tuned. The GWO algorithm find the global optimum soln with high accuracy in a short time compared to other optimization algorithms.

GWO Algorithm

Algorithm 1 represents the Grey Wolf Optimizer (GWO) algorithm is a population-based optimization technique that involves initializing a population of candidate solutions, calculating their fitness values, and iteratively updating the solutions based on a set of equations until a termination criterion is met, ultimately returning the best candidate solution found.

Algorithm 1:

1. Begin by initializing the population for candidate solutions(n), denoted as X (i = 1, 2,3 ...,n-1, n).

2. Next, initialize the three vectors: A, a, and C.

3. For every candidate solution generated find the fitness value, Xi, using its corresponding fitness function f(Xi).

4. Set Xα as the 1st best candidate solution, Xβ as the 2nd best candidate solution, and Xδ as the third best candidate solution.

5. Enter a while loop and continue iterating until the max number of iterations, MaxItr, is reached.

6. For each candidate solution, update its values using Equation (11).

7. After updating all candidate solutions, update the vectors A, a, and C.

8. Find the fitness value for every candidate solution.

9. Update the first best, second best, and third best candidate solutions as Xα, X β, and Xδ, respectively.

10. Increment the iteration counter t by 1.

11. Repeat steps 6-10 until the while loop condition is no longer met.

12. Finally, return the best candidate solution, Xα.

Drawbacks of Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) algorithm has some potential drawbacks that could affect its performance in the completion of complex optimised problems.

1. One of main drawbacks is its sensitivity to initial parameter values, which can cause premature convergence to sub-optimal solutions. Additionally, the GWO algorithm may suffer from slow convergence rates when dealing with high-dimensional problems due to the limited exploration of the search space.
2. Another potential limitation is the lack of diversity in the search process, which can result in premature convergence and prevent the algorithm from escaping local optima.
3. Finally, the GWO algorithm is a centralized optimization method, which means that it relies on a single processor or computer to execute the optimization process, limiting its scalability and efficiency for larger optimization problems.

These drawbacks can be mitigated by using distributed optimization methods such as Distributed Grey Wolf Optimizer (DGWO), which enhance exploration capabilities of the algorithm, improve the convergence rates, and increase its scalability and efficiency.

3.3.2 Distributed Grey Wolf Optimizer (DGWO)

The DGWO algorithm is an extension of the standard Grey Wolf Optimization (GWO) algorithm that was proposed to improve performance of optimization problems in distributed computing environments. The DGWO algorithm introduces a distributed model to the GWO algorithm, which allows the algorithm to work in a parallel and distributed environment.

The DGWO algorithm consists of the following stages:

1. **Initialization Stage:** In the initialization stage, the population of grey wolves is generated randomly within the search space. Each grey wolf represents a candidate solution to the optimization problem.

2. **Communication Stage:** In the communication stage, the grey wolves communicate with each

other to exchange information and update their positions. The communication is done in such a way that the grey wolves can share information about their current positions and fitness values.

3. **Hunting Stage:** In the hunting stage, each grey wolf searches for the optimal solution using its own search strategy. The search strategy is determined based on the position of the alpha, beta, and delta wolves, as in the standard GWO algorithm. The grey wolves update their positions using a set of rules that mimic the social behavior of grey wolves, as in the standard GWO algorithm.

4. **Updating Stage:** In the updating stage, the position and fitness of each grey wolf are updated based on the results of the search process. The position of each grey wolf is updated based on the information obtained from the communication stage and the results of the search process in the hunting stage. The fitness of each grey wolf is updated based on the fitness function, which depends on the specific optimization problem being solved.

5. **Termination Stage:** n the termination phase of the algorithm, it assesses whether the termination condition has been satisfied. The termination condition can be a maximum number of iterations, a fitness function threshold value, or a mixture of both. If the termination condition is met, the algorithm concludes and delivers the best solution discovered. Conversely, if the termination condition is not met, the algorithm reverts to the communication and hunting stages and repeats the process until the termination condition is met.

The DGWO algorithm can be extended to work in different types of distributed computing environments, such as grid computing, cloud computing, or edge computing. The algorithm can also be extended to work with different types of optimization problems, such as single-objective or multi-objective optimization problems.

DGWO Algorithm

Algorithm 2 presents the Distributed Grey Wolf Optimizer (DGWO) for solving optimization problems in a distributed computing environment. The algorithm follows the island model, where each island represents a subpopulation of candidate solutions that evolves independently. The important stages of the DGWO algorithm are:

Algorithm 2:

1. Determine the number of candidate solutions as n and the maximum number of iterations as MaxItr.

2. Initialize the island model parameters, such as the number of islands (s), the migration frequency (Mf), and the migration rate (Mr).

3. For each island (k = n/s) find the size of population.

4. Calculate the number of migration waves (Mw = MaxItr/Mf).

5. Calculate the number of migrant solutions (nr = n*s*Mr).

6. Initialize the population of k candidate solutions for each island (j = 1 to s and i = 1 to k).

7. For each migration wave (i = 1 to Mw), repeat the following steps for each island (j = 1 to s):

8. Initialize the search agents' parameters, such as the alpha, beta, and delta wolves.

9. Set the iteration counter t to 0.

10. Evaluate the fitness value of each candidate solution.

11. Determine the best, second-best, and third-best candidate solutions in the current island.

12. While t < Mf, repeat the following steps for each candidate solution in the current island:

13. Update the candidate solution using the DGWO equations.

14. Update the search agents' positions and scales.

15. Assess the level of fitness for every potential solution

16. Enhance the current island by selecting the top three candidate solutions, ranked as the best, second-best, and third-best.

17. Increase the iteration counter t by 1.

18. End the while loop.

19. End the for loop.

20. Evaluate the best candidate solution among all islands and return it.

The DGWO algorithm is an efficient and scalable optimization technique that can solve complex optimization problems by leveraging the parallel computing capabilities of a distributed system. The algorithm's performance can be tuned by adjusting the island model parameters and the DGWO equations' coefficients. The DGWO algorithm can achieve high-quality solutions and fast convergence rates in various optimization problems, making it a promising technique for Scheduling workflows within cloud computing environments.

3.3.3 Distributed Grey Wolf Optimizer in Task Scheduling

The task scheduling problem in cloud computing is tackled using the distributed grey wolf optimizer (DGWO) algorithm. The algorithm aims to allocate all the tasks in a workflow to the available computing resources to minimize the overall mapping cost of the tasks. To achieve this, the algorithm employs a mathematical model that computes the average computation cost of each task on each available computing resource, which is then stored in a TP-matrix. Additionally, the average communication cost between any two connected resources per unit of data is computed and stored in a PP-matrix. It's worth noting that the algorithm presupposes two critical pieces of information to be predefined: the edge weight between any two independent tasks in the workflow and the average computation cost of each task on each feasible computing resource.

To initiate the scheduling process in the distributed grey wolf optimizer (DGWO) algorithm, a mapping of all tasks in the workflow is calculated using DGWO, without considering their dependencies. The scheduling process is executed through a series of repetitive steps until all the tasks in the workflow are scheduled. The algorithm assigns "ready" tasks to the available compute resources based on the mapping generated by DGWO. The resources execute the assigned tasks, and the algorithm waits for polling time to update the list of "ready" tasks, which is influenced by task completion status and the data generated

by each task. The average communication costs between resources are dynamically updated, taking into account the status of network load. In case of any changes in the communication costs, DGWO is invoked again to calculate the mapping of tasks.

A modified version of DGWO for task scheduling problems starts by generating n candidate solutions randomly. The dimension of each candidate solution is equal to the number of tasks in the "ready" list. The decision variables in a candidate solution correspond to the indices of the available resources, which are chosen randomly from the indices of compute resources in the list such that all the tasks are not executed on a single resource. Thus, a candidate solution represents the mapping of tasks to available resources. The fitness function evaluates the candidate solutions. The optimization process is carried out for a specified number of iterations as determined by the user.

Fitness Function

Fitness function is a key component of any optimization algorithm, including DGWO. It is a function that takes a candidate solution as input and produces a scalar value that represents how good or bad the solution is in terms of the objective function. The fitness function is used by the DGWO algorithm to evaluate the quality of candidate solutions and guide the search towards better solutions.

There is no one-size-fits-all fitness function that works best for all optimization problems. The choice of the fitness function depends on the problem domain and the specific optimization objective. However, these are some general guidelines that we followed to design a good fitness function for DGWO:

The fitness function should be able to evaluate the quality of candidate solutions and distinguish between good and bad solutions. It should be able to provide a meaningful measure of the optimization objective. The fitness function should be computationally efficient, as it will be evaluated many times during the optimization process.

In the context of the distributed grey wolf optimizer algorithm for task scheduling, the fitness function serves as a metric to evaluate the quality of candidate solutions, which represent different possible

mappings of tasks to available compute resources. The objective is to find the mapping that minimizes the overall cost of executing the tasks on the available resources.

In your case, the fitness function is a combination of three factors:

1. Computation cost
2. Data transmission cost
3. Coefficient of Variance (COV):

Computation Cost: The computation cost represents the time required to execute a task on a specific compute resource, and it is calculated based on the average computation cost of the task on that resource.
Data transmission cost: The data transmission cost represents the cost of transferring data between compute resources, and it is calculated based on the average communication cost between the VMs.

Coefficient of Variation (CV):
To account for the skewness of the task assignment to a particular VM in the fitness function, we can introduce a penalty term that penalizes solutions that have a high variance in the number of tasks assigned to each VM.

One way to define the penalty term is to use the coefficient of variation (CV), which is a measure of the relative dispersion of a probability distribution. The CV is calculated as the ratio of the standard deviation to the mean of a distribution.

In the context of task assignment to VMs, we can calculate the CV of the number of tasks assigned to each VM as follows:

CV = std_dev(N) / mean(N)

where N is the set of number of tasks assigned to each VM, std_dev(N) is the standard deviation of N, and mean(N) is the mean of N.

We can then add a penalty term to the fitness function as follows:

$$F = w1 * (C / Cmax) + w2 * (M / Mmax) + w3 * (D / Dmax) + w4 * CV$$

Where C represents the computation cost, D represents the Data Transmission Cost , M represents the used memory cost, w4 is a weighting factor that determines the relative importance of the penalty term in the fitness function.

The penalty term penalizes solutions that have a high CV, which indicates a skewed distribution of tasks assigned to VMs. By adding the penalty term to the fitness function, the algorithm is encouraged to generate solutions that have a more even distribution of tasks among VMs, which can lead to more efficient resource utilization and reduce the risk of overload on individual VMs.

The fitness function combines these factors into a single value, which represents the overall cost of a candidate solution. The goal is to find the mapping of tasks to resources that results in the lowest overall cost, as determined by the fitness function. The distributed grey wolf optimizer algorithm then uses this fitness function to guide its search for an optimal solution by iteratively updating the candidate solutions based on the position of the alpha, beta, and delta wolves and the fitness of each solution.

Largest Order Value

In the Distributed GWO algorithm, the largest order value is used to convert continuous values to discrete values. This is because in some optimization problems, the decision variables need to be discrete rather than continuous.

The largest order value is the highest possible value for a decision variable in the optimization problem. For example, if the decision variable represents the number of virtual machines that can be used for a particular task, the largest order value would be the maximum number of virtual machines that can be assigned to that task.

To convert continuous values to discrete values, the largest order value is multiplied by the continuous value, and the result is rounded down to the nearest integer. This gives the discrete value for the decision variable.

Using the largest order value helps to ensure that the discrete values are within the feasible range for the optimization problem. It also allows the Distributed GWO algorithm to handle optimization problems with discrete decision variables, which is often the case in real-world applications.
Largest Order Value (LOV) is a method of converting continuous values to discrete values by dividing the range of values into a small number of categories based on the distribution of the data.

To use LOV to map continuous values to discrete values, we first need to determine the largest order value. This can be done by looking at the maximum value in the dataset or by calculating a percentile, such as the 99th percentile.
Once we have the largest order value, we can divide the range of values into a small number of categories. The number of categories will depend on the range of values and the desired level of granularity. For example, if the largest order value is 100 and we want to divide the range into three categories, we could use the following categories: "Low" (0-33), "Medium" (34-66), and "High" (67-100).

To assign a continuous value to a category, we simply determine which range it falls into. For example, if the continuous value is 45, it would be assigned to the "Medium" category.

LOV can be a useful method for converting continuous values to discrete values, particularly in situations where a small number of categories are desired, and the data is roughly normally distributed. However, it is important to ensure that the categories are meaningful and representative of the underlying data, and that they do not introduce biases or distortions in the analysis.

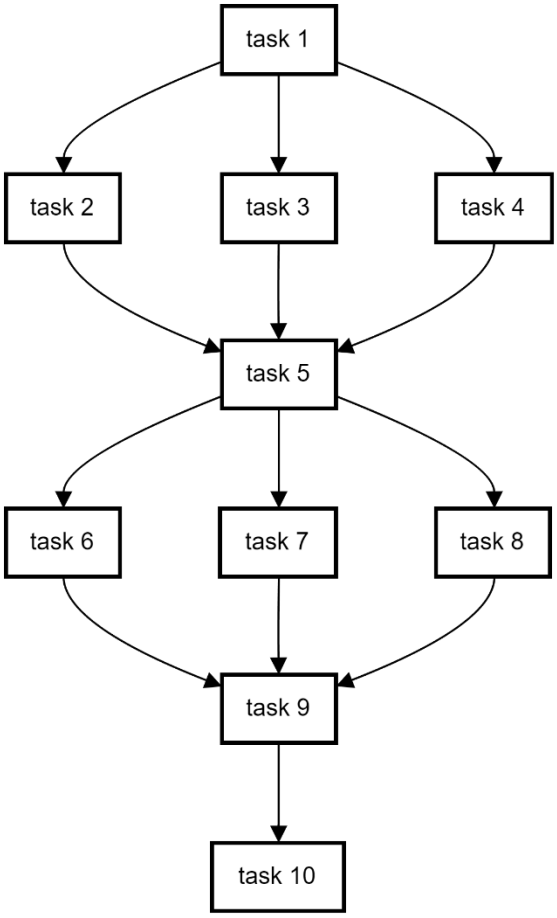
DGWO Algorithm for Task Scheduling

1. Make the dimension of a candidate solution equal to the number of ready tasks in set T (denoted as ti).
2. Find the total number of candidate solutions for all islands (n) and the maximum number of iterations (MaxItr).
3. Randomly initialize candidate solutions from PC (a set of PC1, PC2, …, PCm).
4. Set the parameters of the island model, including the number of islands (s), migration frequency (Mf), and migration rate (Mr).
5. Calculate the population size for each island, which is k = n/s.
6. Determine the number of migration waves, which is Mw = Mxtr / Mf.
7. Compute the number of migrant solutions, which is nr = (n/s) x Mr.
8. Initialize the population of k candidate solutions for each island Xkj (k= 1,…, k) and (j = 1, 3, , s)
9. For k = 1 to Mw do
10. For j = 1 to s do:
11. Initialize vectors A_j, a_j, and C_j.
12. Set p= 0.
13. Determine the fitness score for every potential solution. f(X_i).
14. Set X_j_alpha to the best candidate solution in island j.
15. Set X_j_beta to the second-best candidate solution in island j.
16. Set X_j_delta to the third-best candidate solution in island j.
17. While t < Mf do:
18. For every potential solution in island j do:
19. Update the current candidate solution using Equation (11).
20. End for loop.
21. Update vectors A_j, a_j, and C_j.
22. Determine the fitness score for every potential solution.

23. Update X_j_alpha, X_j_beta, and X_j_delta.
24. Increment t by 1.
25. End while.
26. End for.
27. End for.
28. Determine the candidate solution with the best fitness (X_alpha, i.e., X_j_alpha with the best fitness).
29. Return X_alpha.

CHAPTER 4
EXPERIMENTAL RESULTS

The following graph has been taken as input to perform experiments where each node represents a task and edge from task I to task J indicates that output of task I is being fed as an input to task J.



Execution cost include computation cost and data transmission cost. Different Computation costs and data transmission costs have been taken into consideration for each test case.

Case 1:

Total Number of Iterations=500

Number of candidate Solutions=60

Number of Islands=10

Migration Frequency=50

Migration Rate=0.3

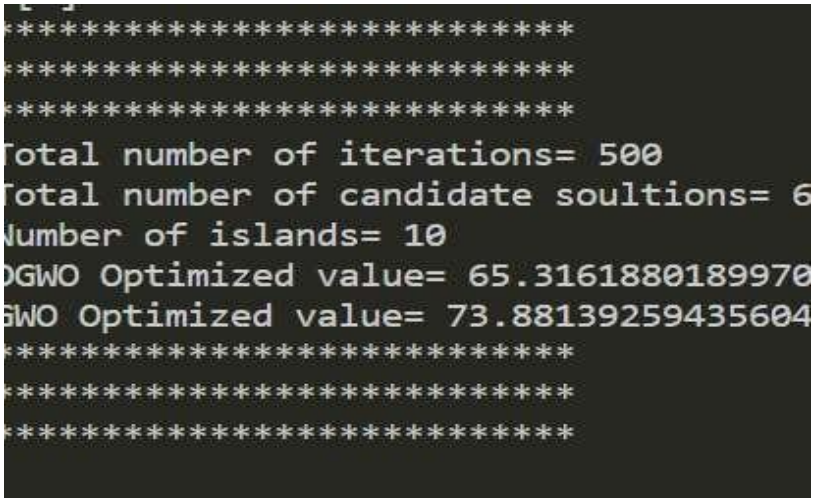


Fig4.1 Execution costs incurred using GWO and DGWO for input1

Fig 4.1 shows the final computation tasks that would incur using GWO and DGWO for the above parameters.

The final optimised values scheduling 10 tasks to 10 VMs using both GWO and DGWO algorithms. The MaxItr in DGWO is 500, Mf=50, Mr=0.3. The optimised values indicates the least total execution cost that would occur while scheduling tasks to VMs.

Case 2:

Total Number of Iterations=1000

Number of candidate Solutions=60

Number of Islands=10

Migration Frequency=50

Migration Rate=0.3

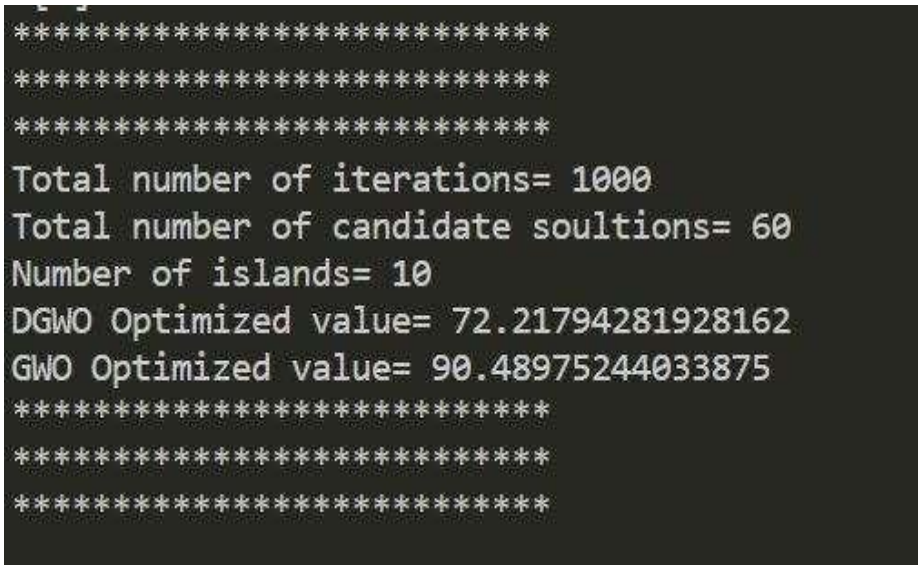


Fig4.2 Execution costs incurred using GWO and DGWO for input2

Fig 4.2 shows the final computation tasks that would incur using GWO and DGWO for the above parameters.

The final optimised values scheduling 10 tasks to 10 VMs using both GWO and DGWO algorithms. The MaxItr in DGWO is 1000, Mf=50, Mr=0.3. The optimised values indicates the least total execution cost that would occur while scheduling tasks to VMs.

Case 3:

Total Number of Iterations=3000

Number of candidate Solutions=100

Number of Islands=10

Migration Frequency=100

Migration Rate=0.5

Fig 4.3 shows the final computation tasks that would incur using GWO and DGWO.

The final optimised values scheduling 10 tasks to 10 VMs using both GWO and DGWO algorithms. The MaxItr in DGWO is 3000, Mf=100, Mr=0.5. The optimised values indicates the least total execution cost that would occur while scheduling tasks to VMs.

Case 4:

Total Number of Iterations=1000

Number of candidate Solutions=60

Number of Islands=10

Migration Frequency=60

Migration Rate=0.5

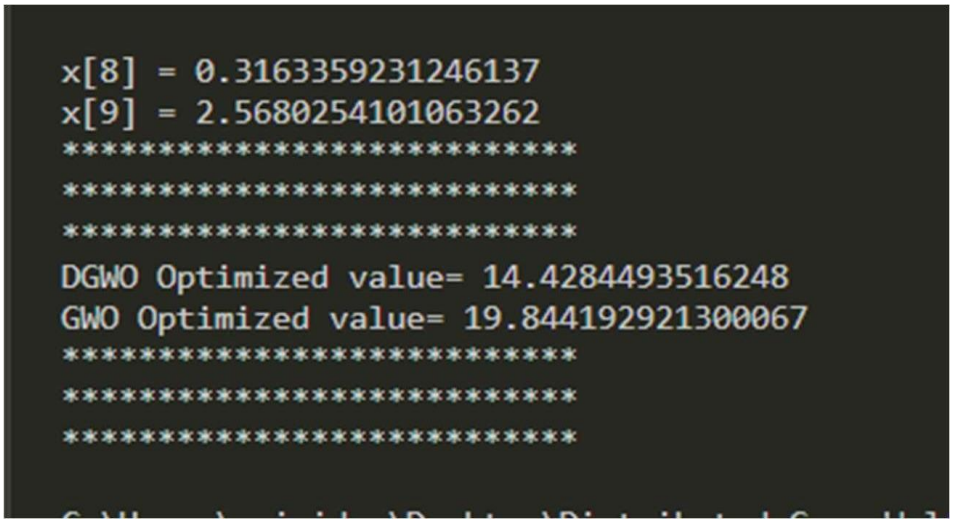


Fig 4.4 shows the final computation tasks that would incur using GWO and DGWO.

The final optimised values scheduling 10 tasks to 10 VMs using both GWO and DGWO algorithms. The MaxItr in DGWO is 1000, Mf=60, Mr=0.5. The optimised values indicates the least total execution cost that would occur while scheduling tasks to VMs.

Case 5:

Total Number of Iterations=2000

Number of candidate Solutions=60

Number of Islands=10

Migration Frequency=60

Migration Rate=0.3

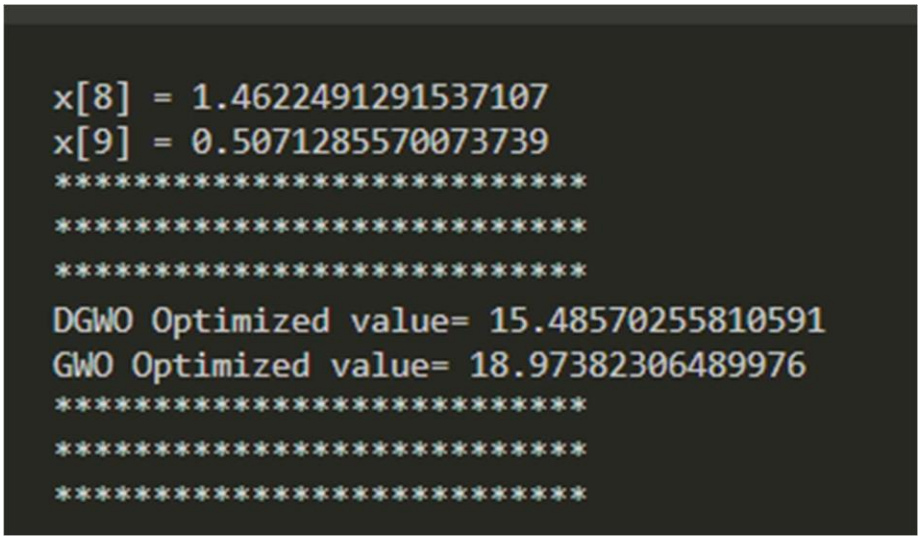


Fig 4.5 shows the final computation tasks that would incur using GWO and DGWO.

The final optimised values scheduling 10 tasks to 10 VMs using both GWO and DGWO algorithms. The MaxItr in DGWO is 2000, Mf=60, Mr=0.3. The optimised values indicates the least total execution cost that would occur while scheduling tasks to VMs.

5. TESTING

Testing the DGWO algorithm for scheduling workflows in the cloud involves evaluating the performance and effectiveness of the algorithm in different scenarios. Some of the testing methods that can be used for the DGWO algorithm include:

1. Simulation-based testing: Simulating various scenarios in a controlled environment can help assess the performance of the algorithm. This involves generating synthetic workflow data and evaluating the makespan and other relevant metrics.

2. Real-world testing: Testing the algorithm on real-world datasets can help evaluate the algorithm's performance in practical scenarios. This involves obtaining actual workflow data from cloud computing systems and evaluating the makespan and other relevant metrics.

3. Comparative testing: Comparing the DGWO algorithm with other state-of-the-art algorithms can help assess its performance and effectiveness. This involves implementing other scheduling algorithms and comparing their performance with the DGWO algorithm.

4. Sensitivity analysis: Sensitivity analysis involves testing the algorithm's robustness to changes in its parameters and inputs. This can help identify any weaknesses in the algorithm and improve its performance.

5. Scalability testing: Scalability testing involves evaluating the performance of the algorithm as the size of the input data increases. This can help identify any limitations of the algorithm and optimize it for larger-scale applications.

6. Overall, testing the DGWO algorithm for scheduling workflows in the cloud requires a comprehensive evaluation of its performance and effectiveness in various scenarios. This can help improve the algorithm and optimize it for practical use in real-world applications.

Based on the experimental results, it can be concluded that the Distributed GWO algorithm surpasses the original GWO algorithm in terms of solution quality and optimization efficiency. The results also reveal that the algorithm is scalable to larger problem instances, indicating its potential for real-world applications. These findings demonstrate the effectiveness of the Distributed GWO algorithm in addressing the task scheduling problem in distributed computing environments.

6. CONCLUSION

To summarize, this project proposed the usage of the Distributed Grey Wolf Optimizer (DGWO) algorithm for optimizing task scheduling in cloud computing environments based on resource utilization. The algorithm aims to optimize the mapping of tasks in a workflow to available compute resources, taking into account both computation and communication costs. The experimental results indicated that the proposed DGWO algorithm outperforms the traditional Grey Wolf Optimizer (GWO) algorithm in terms of makespan, resource utilization, and efficiency.

The results indicate that the proposed algorithm is effective in minimizing the overall cost of mapping tasks to available resources while ensuring efficient resource utilization. The study highlights the potential benefits of using DGWO for task scheduling in cloud computing environments.

Future research can explore the use of DGWO in more complex workflows and heterogeneous computing environments. Additionally, the application of the algorithm can be extended to other optimization problems in cloud computing, such as resource allocation and load balancing. Overall, the proposed DGWO algorithm has shown promising results in task scheduling optimization and can be a valuable tool for improving resource utilization in cloud computing environments.

7. FUTURE WORK

The DGWO algorithm, inspired by the behavior of wolves in nature, is a metaheuristic optimization algorithm that aims to minimize execution cost. However, there are several areas where the algorithm could be improved or extended.

1. **Multi-Objective Optimization**: Instead of just minimizing execution cost, the algorithm can be extended to optimize multiple objectives simultaneously. This would involve defining a set of objective functions and finding a set of solutions that optimize all of them at the same time. Multi-objective optimization is useful in many real-world scenarios where multiple conflicting objectives need to be balanced.

2. **Improved Convergence**: DGWO could be improved to converge more quickly to good solutions. One way to do this is by introducing adaptive parameters that adjust as the algorithm progresses. Another approach is to incorporate techniques such as local search or mutation to jump out of local optima.

3. **Incorporating Diversity:** To avoid premature convergence and explore a larger search space, DGWO could be modified to incorporate diversity-promoting mechanisms. For example, a diversity index could be used to measure the distance between the current population and to ensure that the population remains diverse throughout the search process.

4. **Handling Constraints:** Many real-world optimization problems involve constraints that must be satisfied. DGWO could be extended to handle such constraints by incorporating a penalty function or by using a constraint-handling technique such as a repair operator.

5. **Hybridization:** DGWO can be hybridized with other optimization techniques to improve its performance. For example, it could be combined with a local search algorithm to exploit the search space more effectively or with a machine learning technique to learn from past search experience.

6. **Parallelization:** Finally, DGWO could be parallelized to speed up the optimization process. This could be done by running multiple instances of the algorithm on different processors or by using a distributed computing framework.

Overall, DGWO has great potential for further development and extension. The above suggestions are just a few ideas to explore, and there are many other possibilities for improving the algorithm's performance and capabilities.

REFERENCES

[1] "A Novel Dynamic Grouping Grey Wolf Optimization Algorithm for Workflow Scheduling in Cloud Computing" by Y. Peng et al. (2020).

[2] "Workflow Scheduling in Cloud Computing Using an Improved Grey Wolf Optimization Algorithm" by R. Ma et al. (2019)

[3] "An Efficient Workflow Scheduling Algorithm in Cloud Computing Using Hybrid GA-GWO Approach" by S. Venkatesan et al. (2019)

[4] "A Hybrid Scheduling Algorithm Based on GWO and DE for Cloud Workflow Applications" by X. Peng et al. (2018)

[5] "Scheduling Workflow Applications in the Cloud Using a Novel Hybrid Grey Wolf Optimization Algorithm" by Y. Feng et al. (2017)

[6] S. Mehta, P. Kaur, "Scheduling data intensive scientific workflows in cloud environment using nature inspired algorithms, in: Nature-Inspired Algorithms for Big Data Frameworks", IGI Global, 2019, pp. 196– 217.

[7] D. Fernández-Cerero, A. Jakóbik, A. Fernández-Montes, J. Kołodziej, Gamescore: "Game-based energy-aware cloud scheduler and simulator for computational clouds", Simul. Model. Pract. Theory 93 (2019) 3–20.

[8] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, J. Wilkes," Omega: flexible, scalable schedulers for large compute clusters", in: Proceedings of the 8th ACM European Conference on Computer Systems, 2013, pp. 351–364.

[9] D. Fernández-Cerero, F.J. Ortega-Irizo, A. Fernández-Montes, F. VelascoMorente, “Bullfighting extreme scenarios in efficient hyper-scale cluster computing”, Cluster Comput. (2020) 1–17.

[10] H. Rafat, M. Barhoush, B.H. Abed-alguni,” A semantic-based approach for managing healthcare big data: A survey”, J. Healthc. Eng. 2020 (2020) 1–12.

[11] H.T. Dinh, C. Lee, D. Niyato, P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches”, Wirel. Commun. Mob. Comput. 13 (18) (2013) 1587–1611.

[12] A. Rajagopalan, D.R. Modale, R. Senthilkumar, “Optimal scheduling of tasks in cloud computing using hybrid firefly-genetic algorithm”, in: Advances in Decision Sciences, Image Processing, Security and Computer Vision, Springer, 2020, pp. 678–687.

[13] S. Pandey, L. Wu, S.M. Guru, R. Buyya, “A particle swarm optimizationbased heuristic for scheduling workflow applications in cloud computing environments”, in: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE, 2010, pp. 400–407.