National College of Ireland

Metaheuristic approach of scheduling algorithm to improve execution time in containerized environment

MSc Research Project
Cloud Computing

Naseem Sultana
Student ID: 22152261

School of Computing
National College of Ireland

Supervisor: Dr. Punit Gupta

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Naseem Sultana
Student ID:	22152261
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr. Punit Gupta
Submission Due Date:	14/12/2023
Project Title:	Metaheuristic approach of scheduling algorithm to improve execution time in containerized environment
Word Count:	5355
Page Count:	20

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:	31st January 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):	

Metaheuristic approach of scheduling algorithm to improve execution time in containerized environment

Naseem Sultana
22152261

Abstract

Cloud computing is an elastic, scalable, and cost-effective solution for all organizations irrespective of the nature of their business. Cloud providers like Amazon, Google, Microsoft, and IBM charge their consumers based on the cloud resources they use. Software as a service, platform as a service, and infrastructure as a service are commonly categorized services provided by them. Recently, container as a service (CaaS) has also been introduced to deploy container-based cloud applications to minimize execution time and reduce computational cost, thus increasing the quality of service of the application.

The report explores various algorithms that are used to address optimization problems while deploying containers. Containers can either be deployed directly on the physical machine or data center, or they can be deployed on virtual machines and then physical machines. The report addresses the two-tier placement technique or hybrid virtualization on task placement and execution focusing on container migration, container energy consumption, wait time, start time, and execution time of the VMs. Examining various algorithms on CloudSim, the results in this paper reveal that a combination of container allocation policy, host selection policy, and a metaheuristic algorithm like ACOR (Revised ant colony optimization) helps to minimize the execution time and energy of the systems while working on extended workloads.

keywords: Cloud computing, container, virtual machine, CloudSim, execution time, ACOR

1 Introduction

As technology is evolving, the need for robust, scalable, and secure data processing has always been a priority across the globe, earlier when applications were built on bare machines (Linux or any other OS), it was quite difficult to manage and transport these application files across the companies. That is when the concept of virtualization was introduced where a hypervisor installed would act as an interface and allow various other operating systems to be installed so that applications can execute on those virtual operating systems. However, this seemed a bit tiring job as independent virtual machines needed maintenance.

While the solution to maintain these virtual machines was still being discovered by various developers, the concept of "cloud computing" as an infrastructure was preferred by most organizations. This new concept of cloud computing not only helped the applications to be more scalable and elastic but also focused on providing the service on demand as

per the needs of the consumer. With the increasing demand for scalability and elasticity, the developers had to focus on finding a solution that helped them reduce the overall execution time of the applications across multiple machines.
Each application developed has its dependencies which allow to execute the application without any error, it was not so easy to run the application across various virtual machines at once as it not only increased the load but also increased the cost of the application maintenance on the cloud as it consumed more resources to maintain the health of the application. Containerization of applications was then introduced as a solution to such problems. Containers are lightweight packages that package all the dependencies of the application in nodes and these nodes are placed on a container, an image of the container is then created and placed on various virtual machines or directly on various operating systems. Thus, containers are said to be lightweight when compared to virtual machines. The study of this thesis is based on containers and their orchestration using the existing placement algorithms and finding the most suitable algorithm which helps to decrease the total execution time of the application by implementing meta heuristic algorithms while scheduling virtual machines (VMs) on physical machines (PMs).

1.1 Motivation:

Containers gained immense popularity within a short period due to their ability to build, deploy, and maintain applications easily across the platform. This revolutionized way of transporting the application across various platforms was a dream come true for developers as they had to focus on the dependencies of the applications while deploying. Now the developers just have to package the dependencies using containers such as docker and then deploy it without being much bothered about the host operating system of the end user.
Working with containers is much easier on microservices application architecture which helps to divide the application based on the containers they are going to be placed in. However, the placement of containers on virtual machines is not explored widely. With various container orchestration platforms like Kubernetes, Apache Mesos, and Docker Swarm, developers can manage to orchestrate containers but a definite solution to minimize the execution time is not explored yet as this parameter depends on the various workload that the container is working on.
A couple of research papers have discussed implementing the existing algorithms to improve container orchestrations across the platform, but only a few research papers have compared and combined the existing predefined algorithms. In this thesis, the two-dimensional placement technique is used to place the containers on VMs and then schedule VMs on PMs (Physical machines). CloudSim, a cloud simulator has been used, and the authors of CloudSim have already defined a set of placement algorithms that help place the containers on various VMs. This paper focuses on these predefined container placement algorithms to find out the most effective combination that helps to reduce the execution time of the containerized applications.

1.2 Research Objective:

Scheduling of containers plays a key role in deciding the overall performance of the applications. As mentioned earlier, we already have container orchestration tools available to work on containers. One of the widely used tools is Kubernetes, which is also an open-

source tool that helps package the dependencies of the application and deploy it across the platforms. However, it is important to schedule these containers on VMs strategically to maximize the usage of these tools and minimize the execution time, thus enhancing the quality of the service of the application.

In this thesis we work on the CloudSim simulator, to study various existing algorithms within the simulator and understand how containers can be placed on various virtual machines to minimize execution time. So, the research question of this study is:

How do scheduling algorithms on containers help to minimize the execution time of extended workloads, so that they enhance the overall quality of service of the application on the cloud?

1.3 Target Audience:

This work is for cloud computing students who want to understand the scheduling of algorithms on cloud computing using CloudSim as a simulator. People who have a keen interest in cloud technologies may also find it as a reference to their studies. This might also prove to be useful for people working on DevOps.

2 Related Work

Container clusters can be managed using container orchestration systems, commonly known as container orchestration tools. These tools help to enhance the deployment of the application in clusters from a single interface, thus, speeding up the deployment activities. Apache Mesos, Docker Swarm, and Kubernetes are a few container orchestration tools that help manage container clusters.

Docker Swarm is a user-friendly container orchestration tool that integrates well with Docker. It supports auto-scaling and load balancing, which is well-suited for smaller-scale deployments, where simplicity is prioritized over advanced features.

Apache Mesos is another container orchestration tool that is well known for its resource sharing, multi-tenancy, flexibility, and complexity. This container orchestration tool is designed to be a distributed systems kernel that abstracts memory, storage, CPU, and other compute resources. It allows different frameworks to share resources making it flexible enough to work with, however, due to the requirement of manual configuration the complexity of this tool increases.

Kubernetes is an extensively used open-source container orchestration tool that provides high-scalability solutions for applications irrespective of their complex or extended workloads. The declarative configuration approach makes it easy for the users to specify the desired state of their application.

This section of the report briefly describes how various meta-heuristic scheduling algorithms have been used over the years on different workloads to get optimized results.

2.1 Container Scheduling algorithms:

As the study is based on cloud-based algorithms, the tool used here is Cloudsim, which is used as a cloud simulator tool to work on various cloud-based algorithms. The author Piraghaj et al. (2017) in the research paper on container cloudsim has described briefly how the framework for ContainerCloudSim example has been developed to provide

support for simulations of containerized cloud computing environments. To get the optimized solution for containerized applications the author has defined various algorithms like FirstFit, MostFull, LeastFull, and Random algorithms for container placements and host selection policies as well. In another paper the author Oussama Smimite (2020) has discussed the need for container migrations and highlighted the importance of hybrid virtualization, where containers and VMs can co-exist and interact. The author has focused on various parameters like CPU usage, disk utilization, power consumption, and migration time, the study based on this paper proves that energy consumed by the data center or host is minimal when using containers, as the unused hosts are turned off after migration.

Singh et al. (2014) in his paper has clearly defined the importance of scheduling tasks to find the optimal results. The comparative study of algorithms done by Singh et al. (2014) shows how important it is to schedule the tasks to obtain sustainability, feasibility, and adaptability of the applications in the cloud environment. Lakra and Yadav (2015) has also showcased the importance of task scheduling for improving quality-of-service and optimizing throughput in cloud environments by scheduling tasks in cloud data centers and the need for implementing multi-objective scheduling algorithms. The author has focused on improving the quality of service parameters like cost, resource utilization, and execution time by experimenting with priority scheduling and newer approaches like multi-objective task scheduling on CloudSim.

Scheduling tasks can be challenging while operating data-intensive applications on edge systems due to heavy workload. In the article "Optimized container scheduling for data-intensive serverless edge computing", the author Rausch et al. (2021) explains how serverless computing helps to manage the complexity of decoupling infrastructure by making heuristic trade-offs between data and computing by task-driven simulations in different scenarios. The author has also proposed a scheduler that improves the quality of task placement compared to the state-of-the-art scheduler of Kubernetes.

Author Menouer (2021) has briefly explained how scheduling decisions are taken by the schedulers. Scheduling decisions such as node conditions, pod priority, affinity and anti-affinity rules, and resource availability are considered during the scheduling process. In the research paper, the author explains how a scheduler tries to find the best fit for each pod based on these factors and balance resource utilization.Menouer (2021) also states that default schedulers in Kubernetes is extensible, allowing the developers to implement their own scheduling policies and strategies to make appropriate scheduling decisions.

While there are many cloud simulators, CloudSim is the preferred simulator which has been used extensively by most researchers. A. V. H Sai Prasad1* (2021) has clearly explained how scheduling is performed in CloudSim at the node level, while different scheduling algorithms are implemented at VM level and host level. A. V. H Sai Prasad1* (2021) has used PSO algorithm to improve task assignment problems and schedule jobs in grid computing, as per the results mentioned in the paper, an improvement in 49 seconds of makespan was noted with the integration of the proposed PSO algorithm.

2.2 Various container scheduling algorithms that focus parameters like execution time

Integration of meta-heuristic algorithms like Ant Colony Optimization (ACO) can improve the scheduler's optimality. Kaewkasi and Chuenmuneewong (2017) tried using ACO on SwarmKit, and, a Docker orchestration engine. The results in the paper high-

light that ACO performed approximately 15% better on the same host configuration when compared to the greedy algorithm. Shekar (2019) carried out a similar experiment, where he carried out experiments on kuberenetes by comparing Ant Colony Optimization(ACO) with First Come First Serve (FCFS) and Round Robin in terms of the average response time of tasks on clusters. He demonstrated that using ACO helped to improve the performance of the applications in terms of resource utilization, and reduce makespan and scalability.

Authors Mustapha and Gupta (2024) have proposed a density-based spatial clustering algorithm for task scheduling to achieve a high quality of service in terms of efficiency. The authors have compared the proposed model with ACO and PSO algorithms for task scheduling hence highlighting improvement in execution time, average start time, and finish time.

Meta-heuristic algorithms like ACO have been used to minimize task completion time and improve the utilization of idle resources Rugwiro et al. (2019) has proposed a task scheduling and resource allocation model based on Hybrid Ant Colony Optimization and Deep Reinforcement Learning in cloud computing environments, where task scheduling is performed using a Binary In-order Traversal Tree and the resource allocation is done based on Ant Colony Optimization. Rugwiro et al. (2019) used deep reinforcement learning to reduce space complexity and split resources into state space and action space. His evaluation proved the mitigation of problems related to resource availability and improvement in idle resource utilization.

3 Methodology

This part of the paper discusses various methods that have been followed to experiment, and compare various algorithms, analyze the results, and conclude the findings. Experiments are carried out on CloudSim, a cloud simulator. It is an infrastructure simulator for cloud-based applications, many algorithms have already been defined within the simulator. To follow the two-level optimization techniques, we first focus on container placements. The author Piraghaj et al. (2017) has already described a few algorithms in the simulator. These predefined algorithms are tested in this project to find the most optimal combination of the pre-existing algorithms that helps to execute the real-time applications at the earliest. Energy consumption of the applications is also one of the parameters that is focused on while placing containers on the VM. After finding the most optimal combination of algorithms for containers, the focus is shifted to the algorithms that help VMs to be placed on physical machines or say host systems.

Various container placement algorithms are defined in the CloudSim architecture, Piraghaj et al. (2017) has described many algorithms for containers. He has introduced container placement policies like FirstFit, MostFull, Random, Max Usage and MostCorrelated, which schedule the containers on VMs based on their utilization. Algorithms like Round Robin Algorithm -, MaxUsage Algorithm and MostCorrelated Algorithms are the container selection algorithms based on CPU utilization, energy consumption and load balancing.

Table 1, 2, and 3 are an overview of various container algorithms defined by Piraghaj et al. (2017) in ContainerCloudSim package. The author Piraghaj et al. (2017) has already compared container placement algorithms in terms of container migration and energy consumption in one of his experiments and found the MostFull algorithm as the

optimal one. Shi et al. (2018) and Tang and Meng (2020) have achieved optimal energy consumption by introducing other metaheuristic algorithms like NSGA-II and MaxUsage policy to reduce energy consumption.

Container Placement Algorithm	**Description**
FirstFit	Allocates containers on the first available VM that meets container deployment requirements.
LeastFull	Packs the containers on the least utilized VMs
MostFull	Places the containers on most utilized VMs
Random	Randomly select VMs to place the containers

Table 1: Container Placement Policy Algorithms

Host Selection Policy Algorithm	**Description**
FirstFit	Select an available host that meets the resources
LeastFull	Select the least utilized host
MostFull	Selects the most utilized host
RandomSelection	Selects the hosts randomly from available hosts

Table 2: Host Selection Policy Algorithm

Container Selection Policy	**Description**
MaxUsage	Selects the container with highest CPU utilization for migration to another host.
MostCorrelated	Select the container whose load is the most correlated with the server hosting it.

Table 3: Container Selection Policy Algorithms

In this research paper, experiments have been carried out with various combinations of these pre-existing container algorithms to find the optimal solution in terms of container migration and energy consumption. Further implementation of these algorithms is discussed in the implementation section of the report.

After scheduling the containers, the focus is shifted to scheduling virtual machines on physical machines using various meta-heuristic algorithms. These metaheuristic algorithms help to decide the optimal solution to execute the given workload within the least possible time. This report contains a brief discussion of the meta-heuristic algorithms

used here to compare the algorithms in terms of wait time, start time, finish time and execution time.

Ant Colony Optimization (ACO)

This algorithm is inspired by the natural behavior of real ants, where the ants deposit pheromones concentrations on the ground as a communication channel for the other ants to follow and find their food. It is stated as revised ACOR in the mealpy library after the version of this algorithm was updated. Author,Shekar (2019) has outlined the importance, need, and results of ACO in his research paper. As per Shekar (2019) ACO can be used to solve scheduling problems of virtual machines and cloud computing environments. As a meta-heuristic algorithm, it can be used to address scheduling and optimizing challenges faced in real-world industrial problems.

Guanqaun Wu (2021) proposes an improved ant colony optimization policy to address the shortcomings of the algorithm, the improved ACO uses a scheduling policy to prioritize the completion of minimum tasks thus reducing the starting time of the search. He also introduced a balance factor coordinating local and global pheromones as an updated mechanism and volatilization co-efficient adjustment mechanism, hence improving the global search ability of the algorithm.

Ant Lion Optimization (ALO):

This algorithm is inspired by the hunting patterns of ants and lions, while ants leave pheromones for the other ants to follow, the lion creates a trap to catch its prey. These nature-inspired multi-objective meta-heuristic algorithms help to find optimal solutions by reducing makespan, execution time, and total cost.

The author, A. V. H Sai Prasad1* (2021) has studied the ALO algorithm to ensure the integrity of data in cloud storage adoption, he also experimented by comparing the algorithm with the PSO algorithm and concluded that ALO performs better than PSO in terms of schedule length and system success probability, thus proving it to be an effective approach for optimal scheduling in cloud computing. However, in this paper, we are going to focus on execution time, start time, wait time, and finish time of the applications on cloud infrastructure.

Particle Swarm Optimization (PSO)

PSO, another meta-heuristic algorithm inspired by the nature of bird flocking and fish schooling behavior. As per the study by Al-Olimat et al. (2015), this algorithm does not require gradient information and can be applied to various optimization problems like job scheduling, task assignments, scheduling cost reduction, and energy consumption reduction. It is said to improve and optimize scheduling by minimizing execution time in cloud computing.

The author Alsaidy et al. (2022) has utilized PSO to allocate tasks to VM in cloud computing environments to evaluate task scheduling parameters like energy consumption and total time execution. The authors have compared the PSO with other algorithms to find that the proposed PSO algorithm performs better than other task-scheduling algorithms.

4 Design Specification

This report focuses on two level optimization technique to schedule containers on VMs first and then Vms on PMs, the figure 1 shows the flow of this technique.

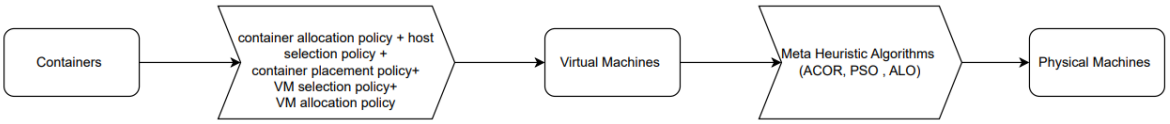

Figure 1: Scheduling Workflow

The examples of Cloudsim are used to implement the technique and find the results. Results and experiments are discussed in the implementation, evaluation, and results section of the report.

4.1 Kubernetes Architecture

As the report focuses on containers, it highlights implementation of Kubernetes as well. Kubernetes (K8s) is a widely used resource management and scheduling system in the cloud. It is one of the most popular open-source container orchestration platforms that automates the deployment, scaling, and management of containerized applications. It consists of a control plane and worker nodes, with various core components responsible for managing the cluster's state. Kubernetes is scalable, highly available, and portable, allowing applications to be deployed consistently regardless of the underlying infrastructure. However, it is complex to set up and operate, requiring expertise and resources.

It provides different scheduling algorithms for scheduling short-running cloud workloads, including artificial intelligence (AI) workloads. The selection of scheduling algorithms has a significant impact on job performance results. However, it is time-consuming to select the optimal algorithm as it takes a few minutes to complete the scheduling process for each job.

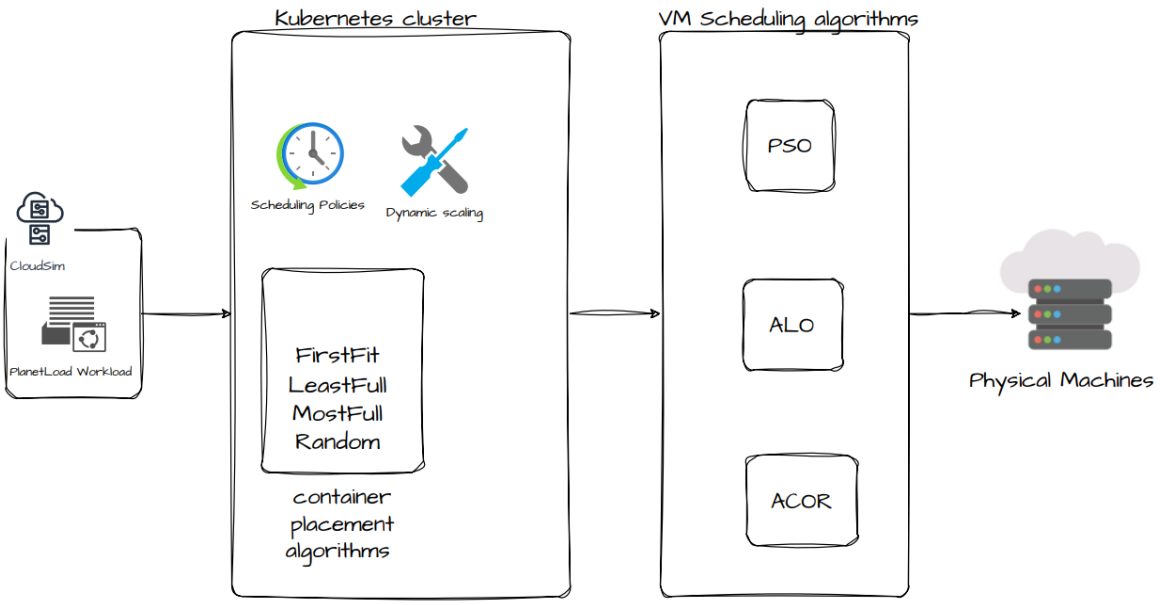

Figure 2: Optimizing Container Orchestration with Hybrid Virtualization

The architecture diagram in figure 2 explains how the implementation has been carried out during the research. The diagram depicts the workflow of the experiments where cloudsim has been chosen due to its ability to model and simulate cloud infrastructure, data centers, and scheduling policies, providing a realistic environment for evaluating container placement and scheduling algorithms. Kuberenetes can be integrated as a container orchestration tool due to its advanced features of container orchestration, including scheduling, scaling, and managing containerized workloads, making it a suitable platform for evaluating container placement policies and metaheuristic algorithms. As a hybrid virtualization model has been implemented in the experiments, only baseline algorithms have been considered for container placements. The selection of specific container placement policies, such as FirstFit, MostFull, LeastFull, and Random, was driven by the need to compare and evaluate different strategies for placing containers on virtual machines (VMs) within the cloud environment. The rationale behind choosing these placement policies lies in their distinct approaches to resource allocation and utilization, allowing for a comparative analysis of their impact on execution time and resource efficiency. After analyzing the impact on containers, scheduling algorithms were implemented on VMs.

Metaheuristic algorithms, such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Revised Ant Colony Optimization (ACOR), were chosen considering their ability to optimize task scheduling and resource allocation in dynamic and complex environments.

By using baseline container placement policies and metaheuristic algorithms in the hybrid virtualization environment, performance metrics such as execution time, resource utilization, and scalability were measured and compared to evaluate their impact on the quality of service and overall efficiency of the cloud environment.

4.2 Proposed Algorithm: Revised Ant Colony Optimization (ACOR)

Metaheuristic algorithms like PSO, ASO and ACOR have been implemented in the experiments during simulation. While a brief description of the algorithms has been mentioned above, a detailed study on the proposed algorithm helps to understand how the algorithms help to provide better results when compared to other metaheuristic algorithms.

Ant colony optimization (ACO) is a bio inspired optimization algorithm that mimics the behavior of ants in finding the shortest path between their nest and the food source. This algorithm has been widely applied to solve NP-hard combinatorial optimization problems, where it has shown promising results. Combinatorial optimization problem refers to a type of problem in which the goal is to find the best solution from a finite set of possible solutions. It involves making choices from a set of discrete elements and finding the combination that optimizes a given objective function. Dorigo et al. (2006) uses the model of a combinatorial optimization problem to define the pheromone model for ACO, indicating that ACO can be modified and applied to solve different combinatorial problems.

The revised ant colony optimization (ACO) algorithm focuses on partitioning artificial ants into two groups: scouts ants and common ants. The common ants follow the searching process of the basic ACO algorithm, where tasks randomly choose resources at each step of solution construction and select the resources to be visited based on a probabilistic decision rule. However, the scout ants have distinct characteristics that set them apart from the common ants. They calculate the resource capacity of the current

optimal solution and search around the optimal solution based on the resource capacity. While the common ants explore resources randomly, the scouts' ants specifically explore resources around the optimal solution, by directly visiting the resources in the optimal solution, rather than relying on probabilistic decision rules. In terms of equation, it can be explained as follows:

$$p_{ij}^{k}(t) = \frac{{\tau_{ij}}^{\alpha}(t) + {\eta_{ij}}^{\beta}(t)}{\sum\limits_{j \in N_t^k} {\tau_{ij}}^{\alpha}(t)}, j \in N_t^k \tag{1}$$

Here N_t^k is the node adjacent to node i.The impact of pheromone concentration can be amplified by α, if N_t^k is large, then it will overpower the impact of pheromone, hence, leads to converging onto a sub-optimal path. Further, when it is $(t+1)^{th}$ iteration of the ant, then the pheromone concentration of each path becomes:

$$\tau_{ij}(t+1) = (1-\beta)x\tau_{ij}(t) + \sum_{n_k}^{k=1} \delta\tau_{ij}^{k} \tag{2}$$

Here, the number of ants is denoted by n_k and the pheromone evaporation parameter by β. According to ACO algorithm, VMs are allocated to the cloudlets in three steps:

Each cloudlet is initialized with special requirements for CPU, memory, and I/O (input/output) resources. These requirements determine the type and number of resources needed for the task to be executed.And the resources are allocated based on three following steps :

- Understanding the user demands
- Initialization of parameters
- Scheduling VMs

Figure 3 illustrates the flowchart of ACO, where each cloudlet is scheduled based on CPU, memory and I/O. The algorithm explains how the cloudlets are initialized and scheduled. Each virtual machine (VM) in the cloud environment has a resource predictor, which is used to search for suitable resources for the cloudlets. The cloudlets aim to find the best solution S_{best} based on their resource's requirements and available resources in the VMs.

Updating Pheromone and Resource Capacity

If the cloudlet reaches the optimal solution, they update the pheromone, which is a chemical substance used by ants to communicate with each other. Then the pheromone update helps in guiding future tasks to find better solutions. Additionally, the cloudlets also search for resources around the optimal solution based on their capacity, which means that they consider the available resources in the Vms and their suitability for the cloudlet execution.

Scheduling cloudlets to resources

According to the revised ACO (ACOR) algorithm, the cloudlets are scheduled based on the available resources. ACOR, takes into account the updated pheromone and the resource's capacity to make better scheduling decisions. The scheduled cloudlets are then shifted into the action space, which refers to the state where the cloudlets are ready to be executed on the allocated resources.

Pheromone and Heuristic Constants

Here, in ACOR, Alpha α and Beta β are considered as the two constants. Alpha α is the pheromone constant, which determines the influence of the pheromone on the cloudlet scheduling process. While beta β is the heuristic constant, which determines the influence of the heuristic information (e.g., resource capacity) on the cloudlet scheduling process.

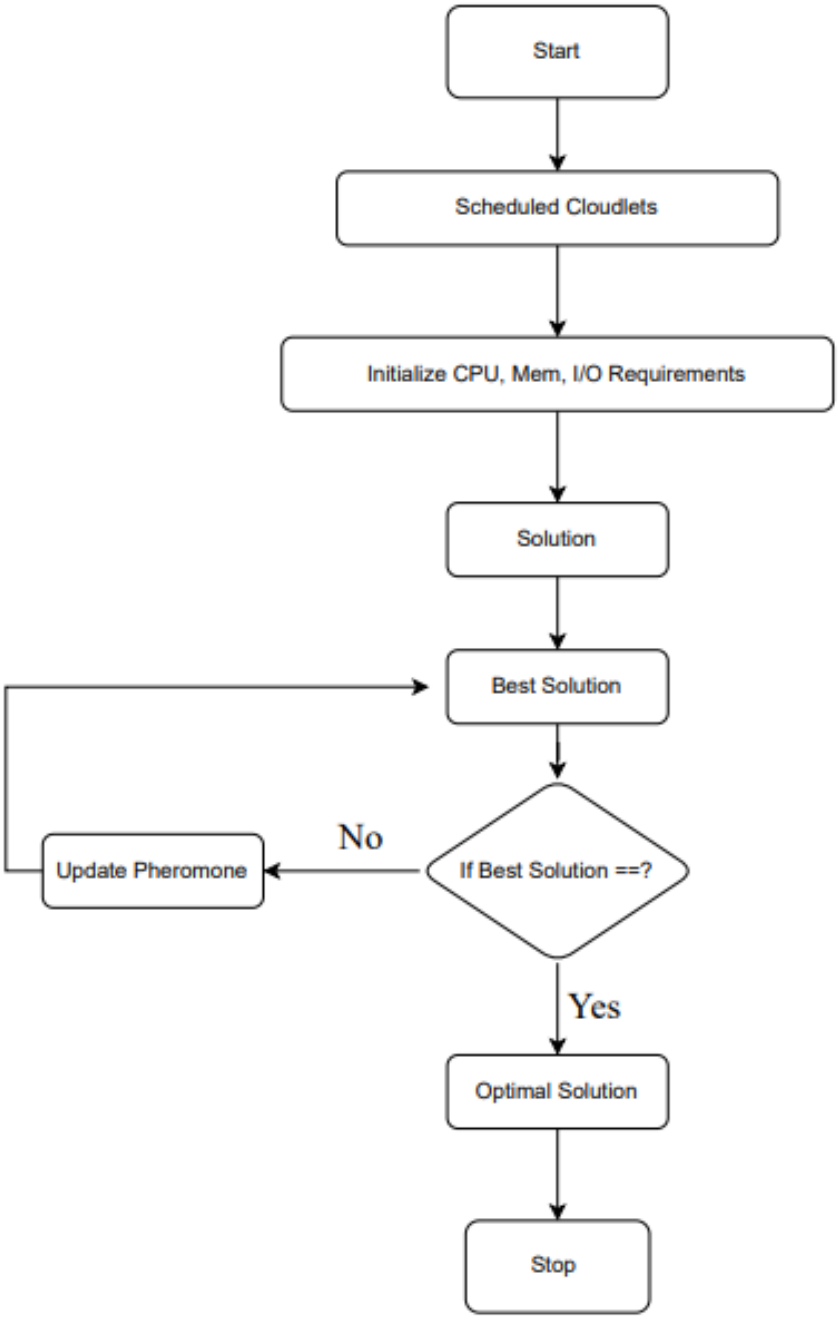

Figure 3: Flow chart scheduling cloudlets

Algorithm 1 : Pseudocode for ACO

Input: Scheduled Cloudlet $T = (T_1, T_2, \ldots, T_n)$
Output: $T_1 \rightarrow R_1$

1. Get $T = T_1, T_2, ...T_n$
2. Initialize $T \rightarrow (CPU, Mem, I/O)$
3. Compute S_{best}
4. Update pheromone
5. If
   $T = T_max$
   Get optimal solution
6. Search the solution around optimal solution
   Else
   Go to step 4
7. End if
8. End

5 Implementation

This section of the report highlights the implementation of the algorithms on the CloudSim simulator. As project focuses on resolving the optimization problem using a two-level scheduling technique:

- Scheduling containers on VMs
- Scheduling VMs on PMs (Physical machines)

CloudSim is used to carry out experiments throughout the project.The pre-existing examples of CloudSim are used as a reference to modify the code as per the requirement and integrate meta-heuristic algorithms. ContainerCloudsimExample.java is the file which is taken as a reference to combine the pre-existing container algorithms, and CloudSimExample6.java is the file on which the meta-heuristic algorithms are implemented.

5.1 Implementation on ConatinerCloudSim to schedule containers on VMs:

Container migration time and energy consumption are recorded by working on ContainerCloudSim files, where a combination of pre-existing algorithms is implemented, compared, and analyzed to find which of the algorithms helps to reduce energy consumption while decreasing container migration time.

```
26
27  //Algorithms
28  public static String VM_ALLOCATION_POLICY = "MSThreshold-Under_0.80_0.70";//[MSThreshold-Under_0.80_0.70, MSThreshold-Over_
29  public static String CONTAINER_SELECTION_POLICY = "MaxUsage"; //[MaxUsage, Cor]
30  public static String CONTAINER_ALLOCATION_POLICY = "FirstFit"; //[LeastFull, MostFull, FirstFit, Random]
31  public static String HOST_SELECTION_POLICY = "FirstFit";//[FirstFit, LeastFull, MostFull, RandomSelection]
32  public static String VM_SELECTION_POLICY = "VmMaxC";//[VmMaxC, VmMaxU]
33  public static String CONTAINER_PLACEMENT_POLICY = "Naseem"; //[Naseem, LeastFull, MostFull, FirstFit, Random]
34
```

Figure 4: Combination of algorithms to find the optimal solution

The above snippet in figure 4 shows a combination of algorithms listed under one file, where each time one algorithm is changed from the list to check the difference in the container migration and energy consumption.

5.2 Implementation on CloudSimExample to schedule VMs on PMs:

Metaheuristic algorithms like ALO and PSO are integrated with the pre-existing CloudSim example to schedule the VMS on PMS and check the execution time, wait, start and finish time each algorithm takes while scheduling VMs to PMs.

6 Experiments and result evaluation

CloudSim 4.0, is used for the simulation of these experiments. The simulation is performed by scaling cloudlets from 100 to 1000 on 10 – 20 VMs. 4 data centers are used in this project with 2 'hosts' each to host VMs. Table 4, refers to the parameters and values used in the experiment for this research. The experiments are carried out by scaling cloudlets to check the performance of the proposed algorithm. The used workload models are of PlanetLab, which represents open-source environment-tailored information for global testbeds Park and Pai (2006).

Parameters	**Values**
Tasks Lengths	500(small) , 1000-2000(medium), 4000(high)
Input file size	300 bytes
Output file size	300 bytes
Fitness function	Execution time

Table 4: Configuration parameters of tasks

6.1 Experiment 1

The first experiment focuses on the container's placements on VMs. The ConatienrCloudSim package is used as a reference to work on combining all the pre-existing algorithms and find the best combination that could take the least time to migrate the containers and reduce energy consumption as well.
With a constant set of parameters, container allocation policy and host selection policies are changed each time to record the results. ContainerAllocationPolicy is a class in the ConatinerCloudSim package that is used to allocate containers to virtual machines. It

selects the available VM in the data center that meets the container's deployment requirements like the container's memory, storage, and availability Shekar (2019). HostSelectionPolicy determines to select an appropriate host for containers based on various parameters like resource utilization, load balancing, and optimization of resource allocation.

According to Boukadi et al. (2017) FirstFit strategy of container allocation policy is not the best in terms of cost when compared to the Linear program, thus increasing the energy consumption indirectly. In this report, algorithms mentioned in container allocation policies like FirstFit, LeastFull, MostFull, and Random are combined with host selection policies to find the optimal results on a set of containers.

This report focuses on finding the combination of strategies that take less time to migrate and consume less energy. A combination of container allocation policies and host selection strategies like FirstFit and FirstFit, LeastFull and LeastFull, and MostFull and MostFull, are recorded and compared by scaling up the containers each time all the combinations are recorded. The results clearly show that the combination of MostFull and MostFull algorithms proved to be a perfect combination as it helped to reduce the container migration time and thus, reduced energy consumption as well.

	Container Migration		
Algorithm	**Lowest**	**Average**	**Highest**
Cloudlets/containers (50)			
VAP = "MSThreshold-Under_0.80_0.70" CSP = "MaxUsage" CAP = "FirstFit" HSP = "FirstFit" VSP = "VmMaxC" CPP = "FirstFit"	300.02	43200.02	86100.02
VAP = "MSThreshold-Over_0.80_0.70" CSP = "MaxUsage" CAP = "MostFull" HSP = "MostFull" VSP = "VmMaxC" CPP = "FirstFit"	**300.02**	**8550.2**	**16800.2**
VAP = "VMThreshold-Under_0.80_0.70" CSP = "MaxUsage" CAP = "Random" HSP = "RandomSelection" VSP = "VmMaxC" CPP = "FirstFit"	300.02	43200.02	86100.02
VAP = "MSThreshold-Under_0.80_0.70" CSP = "Cor" CAP = "LeastFull" HSP = "LeastFull" VSP = "VmMaxC" CPP = "FirstFit"	300.02	43200.02	86100.02
VAP = "MSThreshold-Under_0.80_0.70" CSP = "Cor" CAP = "LeastFull" HSP = "MostFull" VSP = "VmMaxC" CPP = "FirstFit"	300.02	43200.02	86100.02

Figure 5: Container Migration Results

Figure 5 shows the recorded results for container migration. Here combinations of VM allocation policy, container selection policy, container allocation policy, host selection policy, vm selection policy, and container placement policy are executed each time and compared to record the lowest, highest, and average time a container takes to migrate on VMs.

Algorithm	Energy Consumption	
	Maximum	Average
Cloudlets/containers (50)		
VAP = "MSThreshold-Under_0.80_0.70" CSP = "MaxUsage" CAP = "FirstFit" HSP = "FirstFit" VSP = "VmMaxC" CPP = "FirstFit"	10479683.82	5371220.101
VAP = "MSThreshold-Over_0.80_0.70" **CSP = "MaxUsage"** **CAP = "MostFull"** **HSP = "MostFull"** **VSP = "VmMaxC"** **CPP = "FirstFit"**	3588444	1939813
VAP = "VMThreshold-Under_0.80_0.70" CSP = "MaxUsage" CAP = "Random" HSP = "RandomSelection" VSP = "VmMaxC" CPP = "FirstFit"	14105298.02	7705594.48
VAP = "MSThreshold-Under_0.80_0.70" CSP = "Cor" CAP = "LeastFull" HSP = "LeastFull" VSP = "VmMaxC" CPP = "FirstFit"	10547675.7	5614360.51
VAP = "MSThreshold-Under_0.80_0.70" CSP = "Cor" CAP = "LeastFull" HSP = "MostFull" VSP = "VmMaxC" CPP = "FirstFit"	10339044.1	5431161.1

Figure 6: Conatiner Energy Consumption Results

Figure 6 shows the recorded results for container migration. Here the same combinations of algorithms are executed and compared to find the lowest energy consumed while migrating containers to VMs.

6.2 Experiment 2

CloudSim is a cloud infrastructure that provides rich preexisting code to schedule VMs on available datacenter host and perform other cloud simulations. Meta heuristic algorithms are implemented to schedule VMs on available hosts or physical machines (PMs). The result of each algorithm is recorded while keeping the number of VMs constant and changing the number of cloudlets each time. The number of VMs used here is 10 while the cloudlets can be scaled up from 500-2500 to record the results. As mentioned earlier, this project focuses on nature-inspired meta heuristic algorithms: ACOR, ALO and PSO, to find the optimal scheduling algorithm.

The number of cloudlets has been scaled periodically in the code, while Vms are constant and CloudSimExample6.java has been executed to record the results. Each time the number of cloudlets was changed, the meta-heuristic algorithm generated different data sets which showcased the execution time each algorithm was taking to schedule VMs on PMs. Table 5,6,7 and 8, show the results recorded after implementing the experiments. Figures 7, 8, 9 and 10 showcase the results recorded and the graphs plotted. The results in figure 10 clearly depict that ACOR is the most optimal algorithm as it has the least execution time recorded.

Number of Cloudlets	ALO	PSO	ACOR
500	2.08878	3.187398374	3.127656904
1000	2.11795	3.165712831	2.985472689
1500	2.143322215	3.130531697	3.006167832
2000	2.1022	3.196967089	2.985079031
2500	3.195586797	3.195586798	2.989860994

Table 5: Comparison of Average wait time

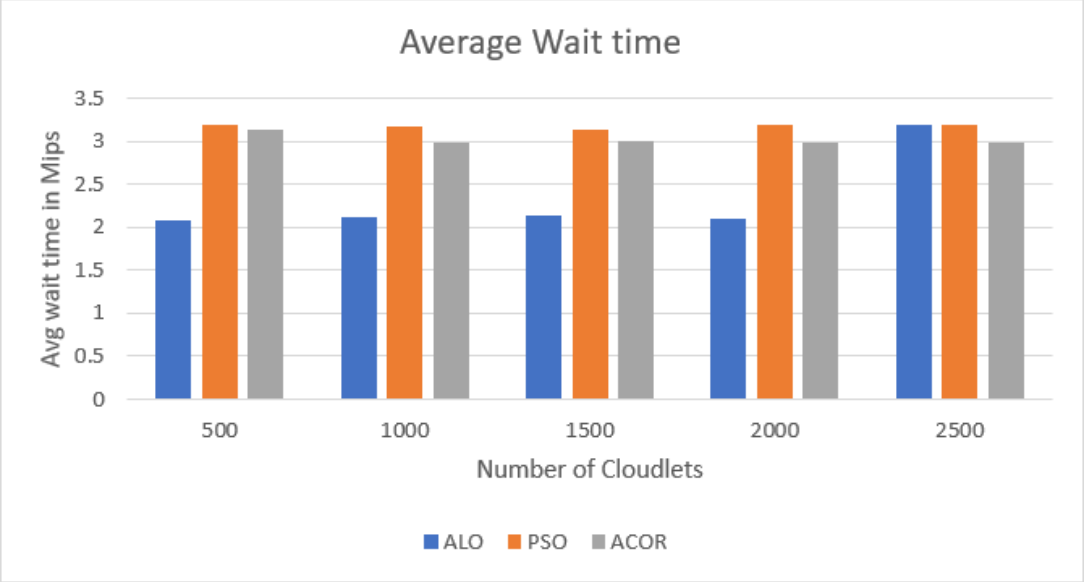

Figure 7: Comparison of Average wait time

Number of Cloudlets	ALO	PSO	ACOR
500	173.82646	116.0661585	103.1658996
1000	310.80462	244.2170876	187.3902626
1500	463.5752168	324.8819836	281.1609441
2000	642.805235	495.2701063	368.3940674
2500	615.7577954	615.7577954	464.1983109

Table 6: Comparison of Average Start time

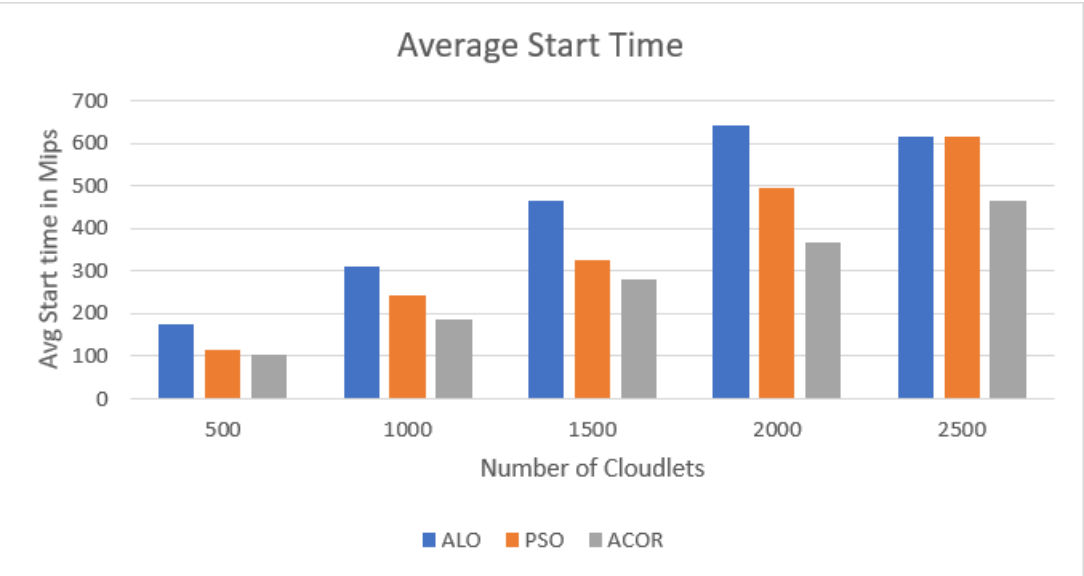

Figure 8: Comparison of Average Start time

Number of Cloudlets	**ALO**	**PSO**	**ACOR**
500	175.91494	119.2528862	106.2929289
1000	312.92205	247.3823218	190.3750105
1500	465.7180254	328.0119155	284.1663916
2000	644.907055	498.4665924	371.3784194
2500	618.9529707	618.9529707	467.1874389

Table 7: Comparison of Average Finish

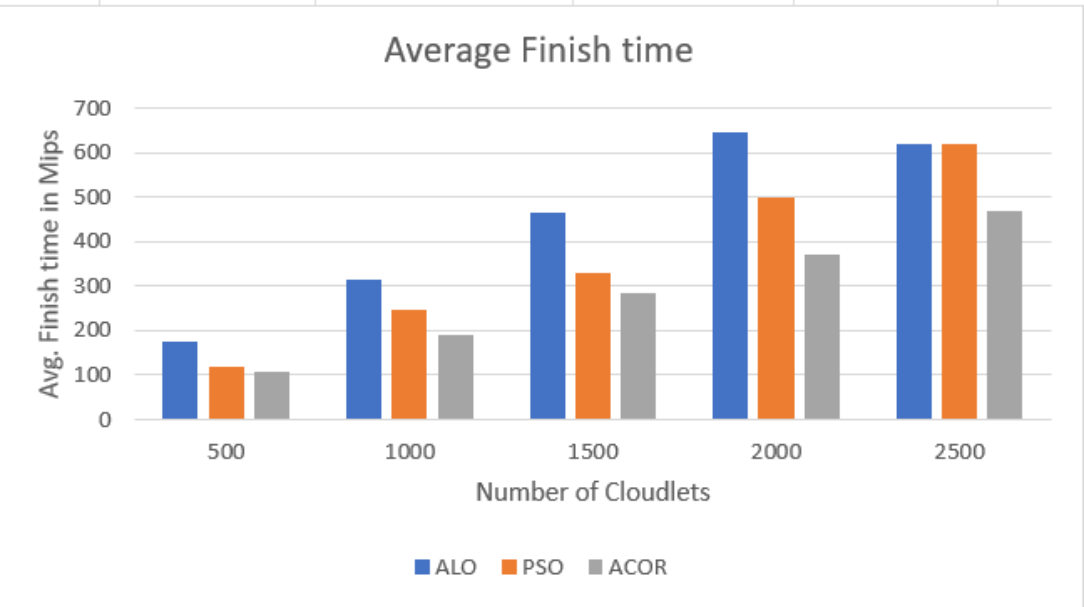

Figure 9: Comparison of Average Finish time

Number of Cloudlets	ALO	PSO	ACOR
500	368.2896399	7.2488451	18.4068195819854
1000	693.0347035	114.447603	60.13135099
1500	1263.156192	24.10668635	128.2013986
2000	830.1434226	51.490799667	136.1294034
2500	1206.308688	236.8184888	157.7574985

Table 8: Comparison of Average Execution time

Figure 10: Comparison of Average Execution time

6.3 Discussion

The experiments and results in the report give a comprehensive exploration of the meta-heuristic algorithms used over the pre-existing container-based algorithms. Examining various aspects of container scheduling and VM scheduling, the results show that PSO and ACOR did not show much difference while recording wait time, start time and finish time. The graphs of these parameters show that ACOR helps to minimize wait time, start time, and finish time. ACOR also proved to be the optimal algorithm in terms of execution time of the application. This study focuses on container migration time, container energy consumption, and execution time. Therefore, based on these parameters, the obtained results suggest ACOR to be the most efficient meta heuristic algorithm which can be combined with MostFull containerAllocationPolicy and Mostfull HostSelectionPolicy to minimize the execution time of the extended workload. The literature review and experiments of this report focus on parameters like migration time and execution time, but the overall performance of the applications depends on various other parameters like security, SLAs and computational cost. These parameters can further be tested by using the same set of algorithms.

7 Conclusion and Future Work

MostFull strategy of container allocation policy and host selection policy has helped to reduce the container migration time and energy consumption, while PSO, the meta-heuristic algorithm helped to schedule VMs to PMs, minimizing the execution time of the application. This concludes that the research has been carried out as per the expectations set initially. The aim was to study cloud infrastructure and experiment with scheduling algorithms which has been fulfilled. However, the research can be extended further by studying the same algorithms on different quality of service parameters like cost efficiency and security. Integration of container orchestrations like Kubernetes and docker swarm can also be considered in the future.
Future works on this research may also include the implementation proposed algorithm (i.e, a combination of MostFull and ACOR) on Amazon or Google cloud using their containers as a service. The algorithm's design and logic dictate how it assesses the current state of the system, evaluates resource availability and makes decisions regarding resource allocation and provisioning. Future work should focus on performance metrics related to scalability and elasticity, such as scalability ratio, resource utilization efficiency, and dynamic resource allocation overhead, to quantitatively evaluate the algorithm's performance in these aspects. By analyzing these metrics under varying workload conditions, the algorithm's effectiveness in handling scalability and elasticity can be systematically assessed on public cloud services like Amazon Web Services (AWS) or Google Cloud Platform (GCP).

References

A. V. H Sai Prasad1*, D. G. V. S. R. K. (2021). A trust model of cloud scheduling based on data integrity using ant lion optimizer, *Turkish Journal of Computer and Mathematics Education* .

Al-Olimat, H. S., Alam, M., Green, R. and Lee, J. K. (2015). Cloudlet scheduling with particle swarm optimization, *2015 Fifth International Conference on Communication Systems and Network Technologies*, pp. 991–995.

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, Part A): 2370–2382.
URL: *https://www.sciencedirect.com/science/article/pii/S1319157820305279*

Boukadi, K., Grati, R., Rekik, M. and Ben-Abdallah, H. (2017). From vm to container: A linear program for outsourcing a business process to cloud containers, pp. 488–504.

Dorigo, M., Birattari, M. and Stutzle, T. (2006). Ant colony optimization, *IEEE Computational Intelligence Magazine* **1**(4): 28–39.

Guanqaun Wu, Rongli Chen, D. Z. X. C. (2021). Using ant colony algorithm on scheduling strategy based on docker cloud platform.

Kaewkasi, C. and Chuenmuneewong, K. (2017). Improvement of container scheduling for docker using ant colony optimization, *2017 9th International Conference on Knowledge and Smart Technology (KST)*, pp. 254–259.

Lakra, A. V. and Yadav, D. K. (2015). Multi-objective tasks scheduling algorithm for cloud computing throughput optimization, *Procedia Computer Science* **48**: 107–113. International Conference on Computer, Communication and Convergence (ICCC 2015).
URL: *https://www.sciencedirect.com/science/article/pii/S1877050915006675*

Menouer, T. (2021). Kcss: Kubernetes container scheduling strategy, *The Journal of Supercomputing* **77**: 4267–4293.

Mustapha, S. D. S. and Gupta, P. (2024). Dbscan inspired task scheduling algorithm for cloud infrastructure, *Internet of Things and Cyber-Physical Systems* **4**: 32–39.
URL: *https://www.sciencedirect.com/science/article/pii/S2667345223000445*

Oussama Smimite, K. A. (2020). Containers placement and migration on cloud system, *International Journal of Computer Applications* **176 - No.35**(5): 1310–1333.

Park, K. and Pai, V. S. (2006). Comon: A mostly-scalable monitoring system for planetlab, *SIGOPS Oper. Syst. Rev.* **40**(1): 65–74.
URL: *https://doi.org/10.1145/1113361.1113374*

Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N. and Buyya, R. (2017). Containercloudsim: An environment for modeling and simulation of containers in cloud data centers, *Software: Practice and Experience* **47**(4): 505–521.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2422*

Rausch, T., Rashed, A. and Dustdar, S. (2021). Optimized container scheduling for data-intensive serverless edge computing, *Future Generation Computer Systems* **114**: 259–271.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X2030399X*

Rugwiro, U., Gu, C. and Ding, W. (2019). Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning, *Journal of Internet Technology* **20**(5): 1463–1475.

Shekar, S. (2019). Improving kubernetes container scheduling using ant colony optimization.

Shi, T., Ma, H. and Chen, G. (2018). Multi-objective container consolidation in cloud data centers, *in* T. Mitrovic, B. Xue and X. Li (eds), *AI 2018: Advances in Artificial Intelligence*, Springer International Publishing, Cham, pp. 783–795.

Singh, R. M., Paul, S. and Kumar, A. (2014). Task scheduling in cloud computing, *International Journal of Computer Science and Information Technologies* **5**(6): 7940–7944.

Tang, L. and Meng, Y. (2020). Energy efficient container consolidation method in cloud environment based on heuristic algorithm, *in* J.-S. Pan, J. C.-W. Lin, Y. Liang and S.-C. Chu (eds), *Genetic and Evolutionary Computing*, Springer Singapore, Singapore, pp. 41–49.