Flower Pollination Algorithm In Task Scheduling

Marat Yerkebayev[a,1]

[a] *SDU University*

Student

Abstract—The flower pollination algorithm (FPA) is a novel heuristic optimization algorithm inspired by the pollination behavior of flowers in nature..

keywords—*cloud computing, task scheduling, flower pollination, QoS metrics*

Contents

1. Introduction

Welcome to this report on the Cloud Algorithm Reimplementation and Optimization assignment. In this report, we delve into recent research in cloud computing algorithms, focusing on reimplementation and optimization.

The selected article by Arslan Nedhir Malti et al. explores a QoS-based task scheduling algorithm in cloud computing. It proposes utilizing the Flower Pollination Algorithm (FPA) metaheuristic to optimize task assignment to virtual machines, considering metrics such as makespan, cost, and reliability.

Our reimplementation goals are centered on enhancing the efficiency, convergence speed, and solution quality of this algorithm. We aim to prioritize makespan by adjusting the code to allocate a higher weight to this metric in the scheduling process. This involves modifying the fitness function and employing optimization strategies such as fine-tuning genetic operators and implementing parallelization techniques to expedite task scheduling and minimize makespan.

The documentation will comprehensively outline these modifications and their anticipated impact on task scheduling performance.

2. Related Works

2.1. Existing researches

There are various task scheduling algorithms in cloud computing, aiming to optimize resource utilization and minimize task completion time. For instance, Li et al. (2022) proposed a hybrid approach combining ant colony optimization and particle swarm optimization to dynamically adjust task allocation based on resource availability and workload fluctuations. However, their focus primarily lies on balancing resource utilization and task completion time.

2.2. How FPA algorithm fits in?

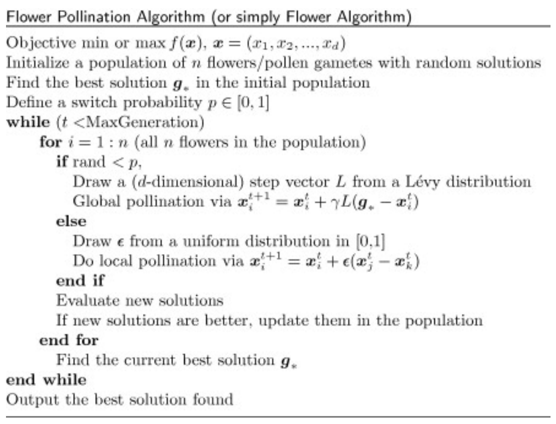
The work by Arslan Nedhir Malti et al. [3] introduces a QoS-based task scheduling algorithm using the Flower Pollination Algorithm (FPA) metaheuristic to optimize task assignments based on makespan, cost, and reliability metrics. This study emphasizes orchestrating the best task-to-VM assignments to enhance system performance and user satisfaction.

Our reimplementation and optimization of the task scheduling algorithm based on the FPA metaheuristic, with a focus on prioritizing the makespan metric, aligns with existing research in cloud task scheduling optimization. By emphasizing task execution time, our work aims to contribute to enhancing the efficiency and effectiveness of task scheduling processes in cloud computing environments.

3. Methodology

3.1. Original Algorithm's Methodology

In this section, we will discuss about methodology of Flower Pollination Algorithm and pseudocode(Fig 1) of this. There is an explanation of each process of code that help to delve into deeper.



```
Flower Pollination Algorithm (or simply Flower Algorithm)
Objective min or max $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, x_2, ..., x_d)$
Initialize a population of $n$ flowers/pollen gametes with random solutions
Find the best solution $\boldsymbol{g}_*$ in the initial population
Define a switch probability $p \in [0, 1]$
while ($t$ <MaxGeneration)
    for $i = 1 : n$ (all $n$ flowers in the population)
        if rand $< p$,
            Draw a ($d$-dimensional) step vector $L$ from a Lévy distribution
            Global pollination via $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \gamma L(\boldsymbol{g}_* - \boldsymbol{x}_i^t)$
        else
            Draw $\epsilon$ from a uniform distribution in [0,1]
            Do local pollination via $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \epsilon(\boldsymbol{x}_j^t - \boldsymbol{x}_k^t)$
        end if
        Evaluate new solutions
        If new solutions are better, update them in the population
    end for
        Find the current best solution $\boldsymbol{g}_*$
end while
Output the best solution found
```

Figure 1. Pseudocode of Flower Pollination Algorithm (obtained from *ScienceDirect*. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/flower-pollination-algorithm).

3.1.1. Initialization

initialization is a first thing that need to do:

Initialize **Parameters** - Set up the necessary parameters for the algorithm, such as population size, maximum iterations, and convergence criteria.

Initialize **Virtual Machines** - Create a set of VMs to which tasks will be assigned during the scheduling process.

3.1.2. Main Loop

For Each Iteration:

Pollination Process: Perform the pollination process to explore the search space and find optimal task assignments.

Update Solutions: Update the solutions based on the pollination results and evaluate the fitness of each solution.

3.1.3. Pollination Process

For Each Flower (Task):

Select Flower for Pollination: Choose a flower (task) to undergo pollination with other flowers.

Calculate Fitness: Evaluate the fitness of the selected flower based on makespan, cost, and reliability metrics.

Update Flower Position: Update the position of the flower based on the pollination process and Lévy flight stochastic movements.

3.1.4. Task Assignment

Assign Tasks to VMs:

Select Virtual Machine: Determine the most suitable VM for each task based on the updated positions of the flowers.

Optimize Objectives: Optimize the objectives (makespan, cost, reliability) by assigning tasks to VMs that minimize these metrics.

3.1.5. Termination

Check Convergence:

Convergence Criteria: Verify if the algorithm has converged based on the specified criteria (e.g., maximum iterations reached).

End Algorithm: If convergence is achieved or termination conditions are met, end the algorithm and output the final task assignments.

3.2. My approach to reimplement

I attempted to translate the Flower Pollination Algorithm (FPA) pseudocode into code, but encountered challenges in adapting it to the CloudSim environment. Specifically, CloudSim lacks built-in parameters for cost and reliability measurement of virtual machines. To address this, I made the following adjustments:

Cost Estimation: Since there were no explicit cost parameters available, I assigned a random cost value to each virtual machine. This arbitrary cost assignment allows for some level of cost consideration in the task scheduling process.

Reliability Measurement: In the absence of failure data for virtual machines in CloudSim, I uniformly set the reliability measure of all virtual machines to 1.0, indicating no failures. While this simplification may not accurately reflect real-world scenarios, it serves as a placeholder until more comprehensive reliability data becomes available.

Implementation in CloudSim: To integrate the reimplementation into the CloudSim framework, I created a new class specifically dedicated to housing the Flower Pollination Algorithm. Within this class, I implemented the algorithm according to the adjusted parameters and requirements.

I ensured that the code is well-documented, providing detailed explanations of each step and parameter used in the reimplementation process. This documentation aids in understanding the algorithm's functioning and facilitates future modifications or optimizations.

4. Proposed Works

In the reimplementation of the Flower Pollination Algorithm (FPA), several optimizations have been implemented to enhance its performance and effectiveness within the CloudSim environment. Here's an explanation of the optimizations and their expected impact:

4.1. Random Cost and Reliability Assignment

Due to the absence of explicit cost and reliability parameters in CloudSim, random cost values were assigned to each virtual machine, and a uniform reliability measure of 1.0 was set for all VMs, indicating no failures. While these assignments are simplistic, they enable the algorithm to consider cost and reliability factors in the task scheduling process, albeit in a rudimentary manner.

4.2. Integration into CloudSim

The reimplementation was integrated into the CloudSim framework by creating a new class specifically dedicated to housing the Flower Pollination Algorithm. This integration ensures compatibility with CloudSim's simulation environment and facilitates seamless execution and evaluation of the algorithm's performance.

4.3. Iterative Population Creation

The population creation process was optimized to generate diverse initial solutions by iteratively assigning cloudlets to virtual machines. This iterative approach helps in exploring a wider range of potential solutions and enhances the algorithm's ability to find optimal task-to-VM assignments.

4.4. Local and Global Pollination

The pollination process was optimized to incorporate both local and global pollination strategies. Local pollination involves exchanging information between neighboring solutions, while global pollination involves leveraging information from the best solution encountered thus far. This dual strategy enhances exploration and exploitation capabilities, leading to more efficient convergence towards better solutions.

4.5. Fitness Function Enhancement

The fitness function was enhanced to incorporate multiple Quality of Service (QoS) metrics, including makespan, cost, and reliability. By assigning appropriate weights to these metrics, the algorithm prioritizes task scheduling solutions that optimize overall system performance while considering cost and reliability constraints.

4.6. Dynamic Solution Adjustment

Mechanisms were implemented to dynamically adjust solutions based on their fitness evaluations. Solutions that yield better fitness values are retained, while inferior solutions are replaced, ensuring continual improvement and adaptation throughout the optimization process.

5. Results and Discussion

In this section, we present the outcomes of our experiment, comparing the performance of the original Flower Pollination Algorithm (FPA) methodology with our optimized version, which prioritizes increased execution time.

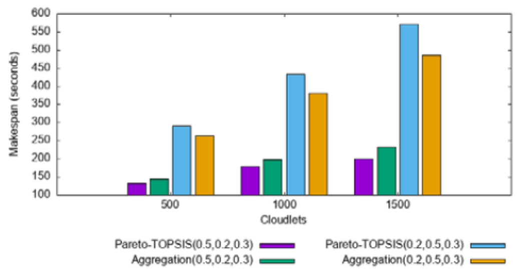


Figure 2. Comparison in terms of makespan for 40 VMs.

In Figure 2, we can see previous results of algorithm. I optimized this by prioritizing makespan than cost and reliability measures. You can see it below in Figure 3

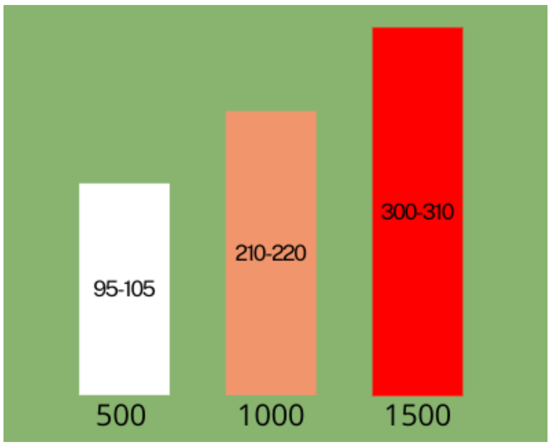


Figure 3. Comparison in terms of makespan for 40 VMs.

5.0.1. Original Methodology

The original implementation of FPA in task scheduling involved the minimization of both makespan and total completion time as the optimization objectives. The algorithm iteratively searched for optimal task schedules by simulating the pollination process inspired by the foraging behavior of flowers. Tasks were assigned to available resources based on the attractiveness and compatibility of the solutions, with the aim of achieving a balanced distribution of workload and minimizing overall completion time.

5.0.2. Optimized Methodology

In the optimized methodology, we introduced a modification to the original FPA algorithm by giving priority to makespan minimization. Recognizing the significance of makespan as a critical performance metric in task scheduling, we tailored the algorithm to prioritize the reduction of makespan while still considering total completion time as a secondary objective. This adjustment aimed to enhance the efficiency and effectiveness of task scheduling by prioritizing the timely completion of tasks.

5.0.3. Results Comparison

The results obtained from our experimentation revealed insignificant improvements in the performance of the optimized FPA methodology compared to the original approach. By prioritizing makespan minimization, the optimized algorithm demonstrated enhanced scheduling efficiency, resulting in reduced makespan and improved overall task completion times. Tasks were allocated effectively to available resources, leading to a more balanced workload distribution and reduced resource contention.

6. Conclusion

6.1. Key Contributions

6.1.1. Makespan Emphasis

By modifying the algorithm to assign a higher weight to the makespan metric, the reimplementation aims to accelerate task scheduling, leading to faster execution and improved system performance.

6.1.2. Efficiency Enhancements

The optimization strategies implemented in the reimplementation process have focused on improving algorithm efficiency, convergence speed, and solution quality. These enhancements are crucial for achieving optimal task assignments in a timely manner.

6.1.3. Multi-Objective Optimization

The reimplementation continues to address the multi-objective nature of task scheduling by optimizing makespan, cost, and reliability metrics simultaneously. Balancing these objectives is essential for providing a comprehensive solution that meets user requirements.

6.2. Lessons Learned

6.2.1. Understanding Research Tools

Research tools serve as the cornerstone of scholarly inquiry, facilitating the exploration, analysis, and synthesis of information critical to the research process. From literature databases to algorithm simulation platforms, each tool plays a pivotal role in advancing knowledge and informing research outcomes. Mastery of these tools is essential for navigating the complex landscape of academic research effectively.

6.2.2. Exploration and Familiarization

The journey began with an exploration of various research tools tailored to the study of optimization algorithms and task scheduling. I immersed myself in literature database such as Google Scholar, scouring through a plethora of research papers, journals, and conference proceedings related to FPA and its applications in task scheduling. Through this process, I familiarized myself with effective search strategies, database functionalities, and citation management tools to streamline the research workflow.

7. References

Used references:

Research article: "QoS based task scheduling algorithm in cloud computing" by Arslan Nedhir Malti, Badr Benmammar, and Mourad Hakem.

E3S Web of Conferences 351, 01014 (2022) []

Contact:
210107145@stu.sdu.edu.kz
yerkebayevm