National College of Ireland

Optimizing Particle Swarm Optimization Algorithm on Serverless Computing for Cost-Efficiency and Performance

MSc Research Project
Cloud Computing

Devashree Shedge
Student ID: x23155906

School of Computing
National College of Ireland

Supervisor: Sudarshan Deshmukh

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Devashree Shedge
Student ID:	x23155906
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Sudarshan Deshmukh
Submission Due Date:	12/12/2024
Project Title:	Optimizing Particle Swarm Optimization Algorithm on Serverless Computing for Cost-Efficiency and Performance
Word Count:	6881
Page Count:	22

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:	28th January 2025

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):	

Optimizing Particle Swarm Optimization Algorithm on Serverless Computing for Cost-Efficiency and Performance

Devashree Shedge
x23155906

Abstract

Serverless computing has enables the rise of the application deployment with cost efficiency and scalability and no overhead in managing infrastructure.But current challenges lies in optimization of resource utilization especially among dynamic workloads. To solve the problems and enhance serverless performance, optimizing it with Particle Swarm Optimization. A novel approach to dynamic optimal resources threshold optimization in introduced through the combinations of PSO to overcome any limitations to traditional optimization techniques. This research deals with train the model,increased resource statistics and strategies of pre-warming. By comparing it with Grid search (Without PSO), PSO is shown to be adaptable to real time optimization tasks. This work represents a valuable insights on the deployment of advanced optimization techniques in serverless architectures. These findings fills the gaps and presents a scalable, cost effective framework for applications across industries like finance, healthcare, e-commerce and logistics.

Keywords: Particle Swarm Optimization(PSO),Serverless Computing, Cost-Efficiency, Performance enhancement,Resource allocation.

1 Introduction

Cloud Computing is the most revolutionary development of technologies since it has made data storage and computing to be flexible, scalable and affordable. Serverless computing, which is represented by tools like AWS Lambda, has became revolutionary allowing customers to run their apps with no need to think about servers. These have greatly influenced various scenarios including the identification cases of financial fraud where real time analyzing is important by reducing potential economic losses. This research is concerned with developing and deploying an effective fraud detection using serverless platforms and an optimum solution such as Particle Swarm Optimization(PSO).
The contribution of this study to existing research is to perform PSO algorithm optimization within AWS Lambda for real world scenarios. To the best of my knowledge, no existing research has been explored the integration of PSO Optimization.

1.1 Background

Machine learning models, usually run constantly on standard servers. However,they are largely resource intensive, expensive and inflexible. Pay as you go and scaling up and

auto scaling capabilities allow for a different solution of these constraints involved in cloud computing with the advancement of cloud computing such as serverless architecture. Researchers had attempted to explore serverless architectures and optimization methods to handle such situations. Busetti et al. [2022] focused on distributed synchronous optimization of PSO for edge computing, in terms of scalability. Furthermore, our work includes a key gap compared with, Delis [2022] work on scaling and distribution of PSO on Microsoft Azure. On the one hand,Delis [2022] research largely concentrates on distributing PSO across configurations on Microsoft Azure system, whereas our work expands this with deploying and evaluating of PSO for a serverless architecture with AWS Lambda. Moreover, this answers the practical questions of real time dataset under concurrent invocation, serverless provisioned concurrency setups and cost performance optimization that were not explored in any of the study till now.

1.2 Motivation

Due to high growth of cloud computing and serverless technologies, application deployment and management have changed rapidly. Problems such as optimal resource allocation, cost effectiveness and achieving high availability in real time did not had any precise solutions yet. Traditional solutions often do not work well in such environments and can be difficult to implement it on large scale. Hence, began with exploring others and similar to PSO, which can enhance performance and cost aspects in serverless systems in a more dynamic manner. This research aims towards offering actionable information that will enhance industries on large and efficient cloud systems.

1.3 Research Question

The focused part of this work is presented as the research question below:

"How Particle Swarm Optimization(PSO) can be combined with pre-warming techniques, resource provisioning strategy and together how will it enhance the performance and reduce costs in serverless computing?"

1.4 Research Objectives

This research, finds out how serverless computing can be optimized in terms of cost reducing and enhancing performance utilizing PSO as a practical framework. This brings a substantial advantages to both the cloud providers and developers as well as for industries with high concurrency requirements, such as e-commerce or financial service sector. It shows real applicability to make scalable, efficient and handle dynamic workloads.

1.5 Structure of this Research

This report is structured as follows: Sections 2 consists of related literature on addressing serverless optimization and PSO techniques. Section 3, describes the preparation of dataset and utilization of the algorithm. In section 4, technical specification and system architecture is mentioned. The implementation process is documented in the section 5 and in section 6, the results are evaluated on the base of case studies of cold start performance, pre-warmed setups and cost analysis. Finally, section 7 concluded with results and future work.

2 Related Work

2.1 An optimization Review of Serverless Computing

Serverless computing had played a major role in evolving cloud architecture by providing cost-effective and scalable solutions to allow different applications.Javed et al. [2024] proposed a comprehensive performance analysis of different serverless platforms which includes OpenFaaS, AWS Greengrass, Apache OpenWhisk, AWS Lambda and Azure functions in resource limited edge environments those are including Raspberry Pi clusters. They tested these platforms for different types of workloads, named as CPU intensive, memory intensive and disk intensive, whereas they found that OpenFaas provides the best latency and performance for edge computing along with that AWS Greengrass faces latency challenges because of its dependency on cloud connectivity. Their study underlines the potential of serverless platforms which also shows the limitations with increasing concurrency, moreover for ARM devices. Alternatively, the survey conducted by Shafiei et al. [2022] in their research they worked on areas like IOT, AI and video such as pay as you go models which works on scalable and cost efficient. This is quite different from where serverless architectures tend to perform relatively well workloads. They struggled with real time applications they require low latency that is a very critical requirement in fraud detection applications. Huang et al. [2023] proposed a PSO-optimized ELM model for short term load forecasting for its further improving the accuracy through optimizing the input weights and thresholds.Their work was used to fit with the random initialization problem in traditional ELM and has better prediction performance. However their results proved the algorithm in simulation but lacked in real world testing. Whereas, Chahal et al. [2021] extended the performnace in terms of cost investigation to AWS Lambda and SageMaker for deep learning based recommendation systems, considering variable workloads. They found that whole AWS Lambda offers better scalability, latency is observed in cold start, while SageMaker performed better under low concurrency situations. Gunarathne et al. [2013] extended an iterative MapReduce framework called as Twister4Azure, which enables large scale parallel data intensive computations over scalability and fault tolerance on the Windows Azure cloud. They mentioned caching mechanism, decentralized scheduling and fault tolerenace strategies for improving performance. Experimental results explains that Twister4Azure performs substantially better than Hadoop for iterative tasks such as KMeans clustering and compared it with Java HPC Twister workload in cloud as well as local environments.

On the other hand, Rostami et al. [2020] proposed a novel feature selection method using node centrality and a multi-objective PSO algorithm to increase performance on high dimensional medical datasets.It optimized to a size, relevance and redundancy heuristic feature subset and integrates it with exploration along with exploitation strategies like initialization by node centrality and mutation operators. The results they got with various filter based and swarm intelligence based methods on a medical datasets.

Although these studies have contributed to serverless computing, it is important as well to consider several limitations of these works. Overall, this point out extremely adaptive and versatile framework for resource allocation, execution time processing and concurrency in serverless architecture. Even though several of this existing methods are scalable, low latency and on serverless environments, which are finding the balance between there factors which mentioned above is challenging. This works fills these gaps by proposing a PSO based framework that allows to work on advance optimization meth-

odology.

2.2 Optimized Cloud Resource Allocation with Advanced Techniques

Several studies worked on modern cloud computing which is highly dependent on good resource management. The techniques which are used in developing techniques in terms of performance, scalability and adaptability.Qawqzeh et al. [2021] carried out a comprehensive study of some algorithms such as Particle Swarm Optimization(PSO),Ant colony optimization(ACO), Artificial bee colony (ABC) and the Firefly Algorithm (FA) they underlined that how precisely they worked on the optimization and high dimensional problems. They concluded that hybrid techniques enhances the scalability and efficiency of heuristics.They faced some challenges while setting the parameters in SI(Swarm Intelligence). Moreover, Pradhan and Bisoy [2022] proposed LBMPSO for task scheduling and load balancing. This was found to give out significantly reduced makespan and in order to improve the utilization of resource. Their work showed the value of dynamic handling tasks and developed efficient systems.The simulations done on CloudSim, it found that LBMPSO works better than other algorithms they performed those are IPSO and LPSO for handling increasing tasks. Shafiei et al. [2022] proposed a hybrid PGA, which are combining PSO and GA. The results showed that compared with the existing methods, it increases resource utilization up to the range of 69.6% and user satisfaction upto 65.4%İn their work, the use of PSO and GA they utilized PSO fast convergence and GA showed how potential it is for hybrid algorithm in solving complex problems. Busetti et al. [2022] investigated distributed Apache Spark and Kubernetes container for optimization in edge computing using 2 PSO variants, such as SDSPSO-LU and SDSPSO-DU. They found that SDSPSO-LU obtained results faster,resulting in a speedup of 5x when it compared with traditional PSO, whereas SDSPSO-DU obtained better fault tolerance. This indeed showed the scalability of PSO in edge environments but also revealed that PSO didnt provided expected results of scalability. Moreover Al-Hashedi and Magalingam [2016], worked on 34 fraud detection techniques on a wide range of fraud domains such as SVM, Naive Bayes and Random Forest. For instance, they discuss the studies of financial domain of fraud from 2009 to 2019 as shown in figure 1. They conducted that SVM was the most accurately fraud detection methodology within bank and insurance fraud due to its accuracy. On the other hand they pointed the challenges in handling imbalanced datasets and adapting to more evolving fraud patterns in which real time solutions has been required.

However, Wang [2010] presented an extensive review of fraud detection methods in telecommunications, insurance and banking using different supervised, hybrid and unsupervised approaches. The results highlighted that hybrid methods are becoming more effective for handling it with new fraud patterns. In contrast of that, they pointed out the severe shortage of publicly available real datasets for doing practical testing, this would allow the validations of practical techniques.

Taking a different approach, Delis [2022] conducted an optimization study for PSO on Microsoft Azure.The author experimented some parameters such as pre-warming instances and activity concurrency and achieved precise drastic reduction of the execution time, along with that improved parallel task distribution. Their study focused only on Azure and it is limited on this platform. Interestingly, they mentioned explore these optimization techniques on other cloud platforms in their future work, this closely aligns

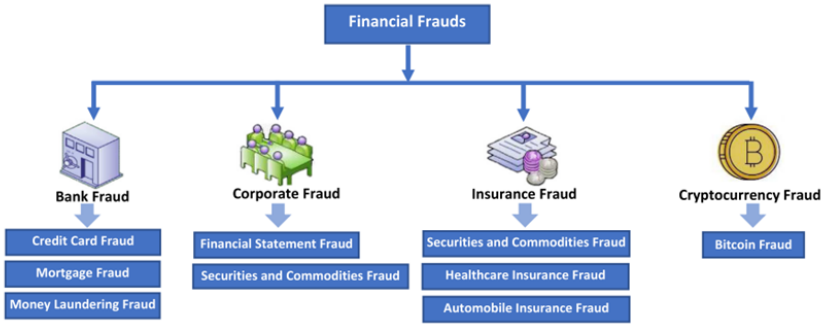


Figure 1: Financial Frauds Domain

on what my research focuses on.

In this section, the reviewed studies described how hybrid algorithms and PSO has been applied on resource allocations,optimization, task scheduling and predicting models on cloud and edge environment. The findings showed how it is scalable in terms of model, increased user satisfaction and greater efficiency in fraud detection improved. However, there are certain gaps, many approaches are based on simulation environments, with an realistic approaches on particular platforms, such as Microsoft Azure. Second, while there is an improvement in fraud detection techniques, not too many studies have combined these methods into serverless architectures for real-time, cost effective optimization. This paper gives the extend insights of the above studies by implementing PSO in a serverless platform for fraud detection.

2.3 Study of PSO Evolution and Real-world Implementation

The study of PSO by Eberhart and Kennedy [1995] was one of the milestone in the history of the development of optimization algorithms. Their algorithm was aimed at modeling the interactions in social behavior of bird flocking, fish schooling and swarm theory. At that time which published in 1995, it gave an effective novel solution to the non linear optimization issues. With this,it gave an effective mechanism of iterating positions and velocities based on an individual and swarm best solutions. Their work of PSO Algorithm was one of the robust optimization techniques is using till now. Their work was further extended by Abbaszadeh et al. [2020], where PSO was applied for the optimization of the parameters for SVM in geological modeling. They classified accuracy which was greater than 97% for mineralization and alteration zones. Their study further emphasized the point that PSO is efficient in faster convergence with higher precision compared to the traditional Grid Search method and hence it got preferred in most large scale classifications. Based on this, Manasrah and Ali [2020] proposed a hybrid GA-PSO algorithm for cloud computing workflow scheduling. This approach combined mutation and crossover capabilities of Genetic Algorithm with the fast convergence of PSO and achieved better reductions about 16% and enhanced in cost efficiency up to 13% this showed the capabilities of hybrid optimization in resource management.Taking a different approach of PSO's, in healthcare domain Choubey et al. [2019] introduced the PSO's applications while integrating it with the kernels of SVM for diabetes classification. Their hybrid framework optimized feature selection and kernel parameters with he accuracy of 79.57% on the Pima Indian Diabetes dataset and 94% on a localized dataset, while computation time got massively reduced. In this direction, Neha and Kumar [2020] applied PSO-SVM

models to measure performance with PCA,focused on datasets with GA.In their case, experimental validity decreases the chance that the methods efficiency in a wise range of real life scenarios has been decreased. In addition, PCA used for features extraction which may not be that helpful on handling the problems with complications on reduction based. This brings out the more requirement for the further study to improve its performance. While analyzing, they reveal that PSO is capable of enhancing the accuracy of classification of the types of data, hence it is efficient and compatible with any type of data. Resolving issues regarding to unstructured dataset, Chen et al. [2016] introduced SMOTE-PSO, a hybrid method is combining synthetic over sampling with PSO. SVM's decision boundaries to get improved metrics such as AUC and G-mean. Their work outlines PSO's potential in reducing biases along with that it, improved the representation in machine learning. Kumar et al. [2023] used Grid search in their work for optimizing the parameter of Random Forest on educational data mining and reported an impressive accuracy of 95.21%.Their interest in minimizing false negatives underlines the importance of predictive accuracy for real world applications like the identification of students at risk.

In this regards, Eberhart and Kennedy [1995] work has inspired a lot of interest in using optimization techniques in different areas. Many studies showed that PSO can enhance the managing resources and scalability. However most of these methods are focused on batch processing. This research paper builds on the ideas by using PSO in a serverless framework to detect fraud in real time, with a focus on better resource utilization and low cost on cloud environments.

2.4 Findings from Related Work

While AWS Lambda optimization has largely been improved, there is a scope of PSO optimization with other latest techniques. Comparisons on various datasets scenarios of various cloud platforms are yet another untouched but potential informative area of research. The related work showed the gaps in theoretical as well as practical implementation. My study aims at filling these gaps by comparing various optimization techniques, performing on experiments and working with performance analysis. It is expected that this work will set a new benchmark for most practical scenarios on how to make AWS Lambda more efficient and cost effective.

3 Methodology

This section describes the methodology which has followed for this study, the research process,techniques of evaluation and various steps from data collection to analysis. The goal was to evaluate the effectiveness of Particle Swarm Optimization(PSO) in enhancing cost efficiency and performance in a serverless fraud detection system using AWS Lambda. Moreover, sections will be describing the procedures of dataset preparation, feature engineering, algorithm implementation and testing some scenarios.

3.1 Research Overview

This research is formatted in different stages such as dataset collection and preparation, algorithm implementation and optimization to provide the detail understanding of this study. These optimization strategies were mentioned related work in 2, Rostami et al.

[2020],they worked on multi-objective PSO algorithm to optimize the high dimensional datasets, along with that they have focused on feature selection for medical datasets, this work generalize the PSO usage for serverless platform, this is not been implemented before.

3.2 Tools and Equipments

This section includes the description of the tools and equipments used in this research as per shown in 1.

Coding Language used	Python 3.9
Development Tools	Google Colab, AWS Cloud9, Visual Studio
Cloud Infrastructure	AWS Lambda, AWS S3, AWS EC2
Monitoring and Testing	AWS CloudWatch, Postman

Table 1: Tools and Equipment Used

3.3 Dataset Preparation

3.3.1 Dataset

The dataset in this research was synthetic and downloaded from data.world [1], a trusted platform for sharing and accessing datasets.This dataset,which was made especially for testing fraud detection models, was downloaded from a trustworthy source. Fraud (isFraud = 1) or non-fraud (isFraud = 0) was assigned to each transaction.

3.3.2 Data Cleaning

After fetching the raw dataset,started with data cleaning on Google Colab[2]. To find out and manage if any missing values is present or not, before doing training and testing the data. Next, used a method to replace this missing values in numerical columns with the average value. This explained that our dataset is not having missing data after performing this step, well it is ready for further analysis.

3.3.3 Feature Engineering

Feature engineering was used to make dataset's more adaptable. This involved :

- One-Hot Encoding : This is a process of converting categorical variables such as the type of branch or transaction, into numerical values represented by binary columns which makes easy to merge it with machine learning models. The model analyze the data and make it appropriate connections between the categories for better understanding.

- In addition to that, the logistic regression model was built and tested with an accuracy of 0.931 which means 93.10% on the maintained dataset.

[1]Raw dataset: `https://data.world/`
[2]`https://colab.google/`

3.3.4 Dataset Storage

Once data pre-processing gets handled, the final formatted dataset was placed on AWS S3 bucket. Storing it in S3, it made convenient to process with next steps. It ensured,that data is safe.

3.4 Algorithm Selection

Shao et al. [2023] found that the hybrid of PSO-GA enhances the efficiency of resource allocation and user satisfaction in optimization. In our work we are utilizing Grid Search(Without PSO) to test a search method which allows varying configurations, although it is computationally demanding.

3.4.1 Grid Search(Without PSO)

- I have initially used Grid Search,which systematically tested a combination of the features of amount, oldbalanceOrg and unusuallogin.
- The error rate was calculated for each combination based on FN and FP and added for thresholds that were set too high. while the logic was straightforward,many combinations needed to be evaluated which made this efficiently possible.
- It worked by trying all possible combinations within a certain range also its looking for the one that performed the lowest percentage error.

3.4.2 Particle Swarm Optimization(PSO)

- Advanced optimization technique like PSO were used to explore the best solutions dynamically.
- How this algorithm works at random intervals of particles with random velocities and its position.
- Each of the particles updated its position in every iteration,this is based on the personal best performance and the global best solution so far.
- While this is an iterative process, thresholds will improve by gradually updating it over the time to minimize the error rate.
- The Fig 2, algorithm explains how the process of PSO algorithm is actually work, how particles improves over iterations and gets initialized. This helps to directly link the implementation of our research and as well as it helps to understand how PSO was applied to optimize the fraud detection of this research.

3.5 Testing

The PSO and Grid Search algorithms are both tested with different memory allocations and variations of the number of iterations and particles respectively. Provisioned Concurrency were implemented and tested to set various experiments as well as the impacts of cold-start times and overall execution performance were also presented.

```
1. Initialize particles (Xi), velocities (Vi), personal best positions (pbest), and global best position (gbest).
2. Randomly generate particles (P) within defined bounds.
3. Evaluate each particle's fitness using the fraud_detection_objective function:
    a. Calculate error rate for each particle.
    b. Update pbest for each particle if the current fitness is better.
    c. Update gbest if the global best fitness improves.
4. For each iteration:
    a. For each particle:
        i. Update velocity (Vi) using:
            Vi = ω * Vi + c1 * r1 * (pbest - Xi) + c2 * r2 * (gbest - Xi)
        ii. Update position (Xi) using:
            Xi = Xi + Vi
        iii. If Xi exceeds bounds, clamp Xi within limits.
        iv. Recalculate fitness and update pbest and gbest.
5. End iterations when max_iterations is reached.
6. Return the scaled_best_solution and gbest_score.
```

Figure 2: Algorithm of PSO

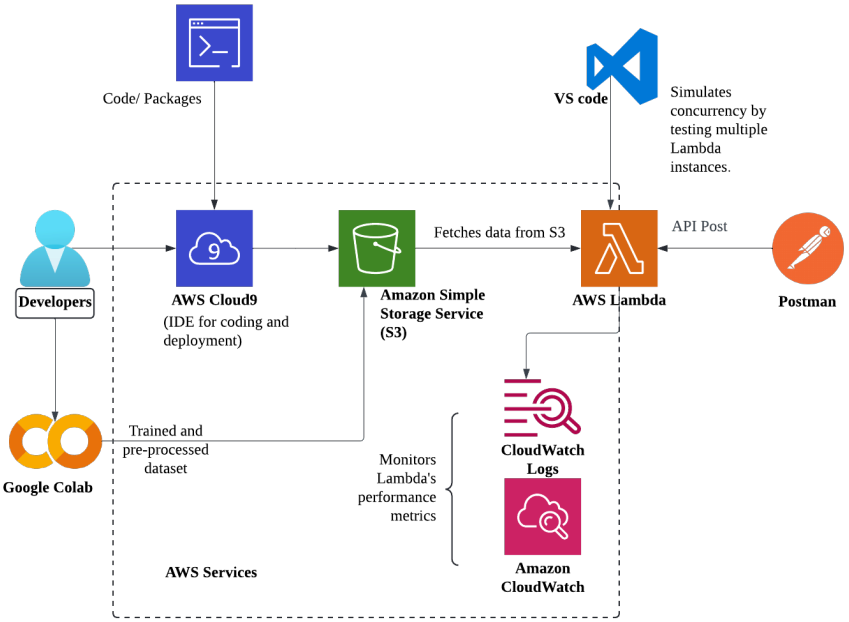


Figure 3: Architecture diagram

4 Design Specification

This section presents the structure,methods and the architecture which will endorse the research of the fraud detection using PSO optimization algorithm in the serverless environment. The aim is to develop a scalable and reasonable solution for AWS Lambda, which allow to maximize the performance and utilization of resources. The following sections describe about the implementation part of the work.

4.1 System Architecture

Fig 3, shows the architecture design of our research. Which has been set up the main aim for utilizing AWS services in such a way, so that it will leverage serverless computing. The architecture of this work, about its various components is explained below in detail.

- Google Colab : Started by using Google colab for pre-processing the raw dataset. Data cleaning, missing values and feature engineering was done to normalize the dataset. Then AWS S3,the data is stored to prepare for further analysis.

- Amazon S3 :In S3 bucket, stored a preprocessed dataset in CSV format. It is easily accessible to fetch the data for aws lambda.

- AWS Cloud9: AWS Cloud9 is utilized for creating, debugging and deploying including python libraries. It allows for seamless development environment for testing and deploying purpose.

- AWS Lambda: The main execution is performed in this aws service, in which both PSO and without PSO algorithms are implemented. Used two separate lambda functions in our research. This helped to compare both the algorithms in a precise way.

- Visual Studio Code: VS code was used to automate the calling of the lambda functions.This helped in effectively triggering a large number of instances and the results will get generated. This allowed to run the code in multiple instances at once, it ensured that the function performed well.

- Postman : It enables to call API for testing purpose on lambda function, also it verifies the deployed algorithm is working properly or not during the testing time.

- AWS CloudWatch : AWS CloudWatch monitors the logs from the execution of lambda function this includes: errors, execution time and outputs. It also records lambda invocations memory size utilization, bill duration and initialization time to make sure system works well.

4.2 PSO Functioning

In this study, PSO algorithm plays an crucial role in optimizing thresholds using transaction data. PSO has been adopted for its ability to converge efficiently on optimal solutions. Compared this algorithm with Grid Search, to figure it out which algorithm works better. Detailed explanation of this functionality is mentioned below:

- Initialization of Particles : The PSO algorithm starts with initializing a particles. Each particle represents a solutions, in terms of set of threshold values for each important features in the dataset those are for: amount, oldbalanceOrg and unusualLogin. The features were chosen based on the importance of the task of fraud detection since those directly defined for fraudulent transactions. The PSO algorithm places each particles inside its define range, from 0 to 1 randomly. It will help to normalize the features. Thus the thresholds values for each particles are randomly assigned from the normalize data range.

- Function : The objective function, fraud_Detection_objective it compares each particles position by its error rate. It find the error rate by comparing the threshold-based fraud predictions of a particle against the actual labels of fraud from the dataset.It will reduces the false negatives more precisely to prioritize actual error in fraud detection.

  The research purpose is to find out the optimal set of thresholds of the fraud detection which is the best solution of the issue and leads to minimize the error rate. The best score is the lowest error rate obtained during optimization process and this is the clear indication of an algorithm's effectiveness.

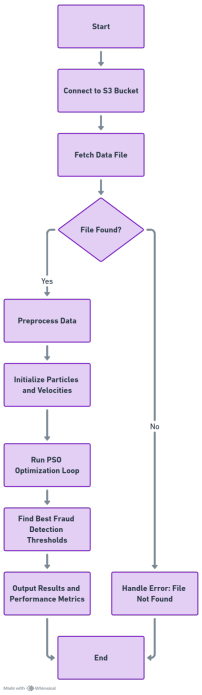


Figure 4: PSO Workflow

There are three variables that impact the velocity of a particle in this formula:

*velocities = 0.5 * velocities + r1 * (personal_best - particles) + r2 * (global_best - particles)*

- Inertia, personalbest and globalbest: While r1 and r2 are random numbers between 0 and 1.This introduces the randomness into the search,whereas inertia termed as 0.5 to maintain the particles current direction. Both the social part attracts the particle towards the globalbest are controlled by unexpected parameters as mentioned in the work by Shi and Eberhart [1998]. In our work, we skipped the coefficients c1 and c2 to simplify the model for its optimal solution.Well,its focus on the algorithms simplicity and efficiency.

- PSO Workflow : As mentioned above, data was preprocessed then created a connection to AWS S3 bucket and stored the dataset over there. It does normalization of the dataset in the next stage of data pre processing if the data is already there presents in s3 bucket. It will then calculates the best fraud detection thresholds and initialized the particles in the PSO algorithm performs the optimization loop once the data is ready. Best solution and best score along with the performance metrics are output as the algorithm saves that for further analysis. All these bring about efficient and accurate threshold optimization that allows for fraud detection. Please find the below workflow diagram 4.

4.3 Configuration Requirements

Serverless deployments is a fundamental aspects, because of its scalability and cost-effectiveness. Combining various AWS services those are: AWS cloudWatch for monitoring purpose, AWS S3 for storing the csv file and AWS Lambda for computing.The

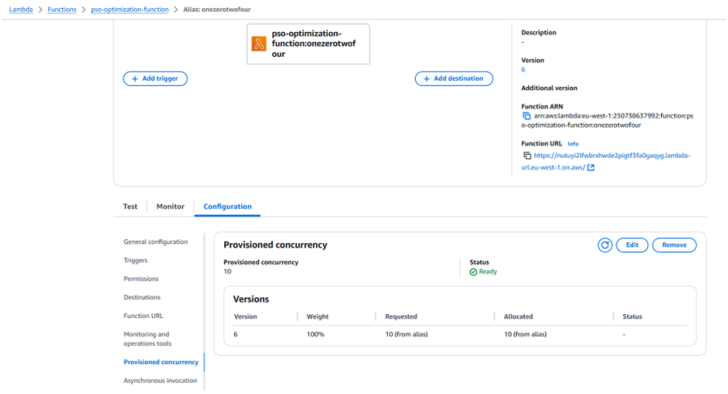


Figure 5: Assigned Provisioned Concurrency

serverless architecture ensures a reduction for an unnecessary infrastructure, at the same time it will ensure that system can handle dynamic workloads. And how it will give best results to a developers/users.

4.3.1 AWS Lambda Configuration

- Memory Allocation: To evaluate systems performance and cost,testing has been done with various configurations of memory.

- Provisioned Concurrency : It sets a number of Lambda instances to warmed up and to get ready[3].Created an alias with latest version of functions. Most importantly,it offers consistent execution performance and faster cold start for reduce the latency.

  For instance, in our case to able to relate it to the specific version of the lambda function, as shown in the 5 fig, created a new alias to linked it with specific Lambda function version. The provisioned concurrency for this function had been set to 10, so that only 10 instances will be available to run the function every time as a pre-warmed.

- Versions and Aliases : AWS Lambda versioning is utilized to ensure that updates in a way that will prevents the disruptions. Created an alias pointing to the latest version of the function.

4.4 Monitoring and Logging

In order to evaluate the performance of the Particle Swarm Optimization (PSO) algorithm, utilized AWS CloudWatch to observe the most important performance parameters at runtime. It showed the execution time, memory used, billed time and the success rates were obtained from CloudWatch and it shows how the algorithms worked on different configurations.

Execution metric : The output shows the clear evidence that PSO performance improved, in fig 6,also confirmed that execution times reduces when memory is increased which proves that resource allocation is critical for gaining best performance.

[3]`https://aws.amazon.com/blogs/aws/new-provisioned-concurrency-for-lambda-functions/`

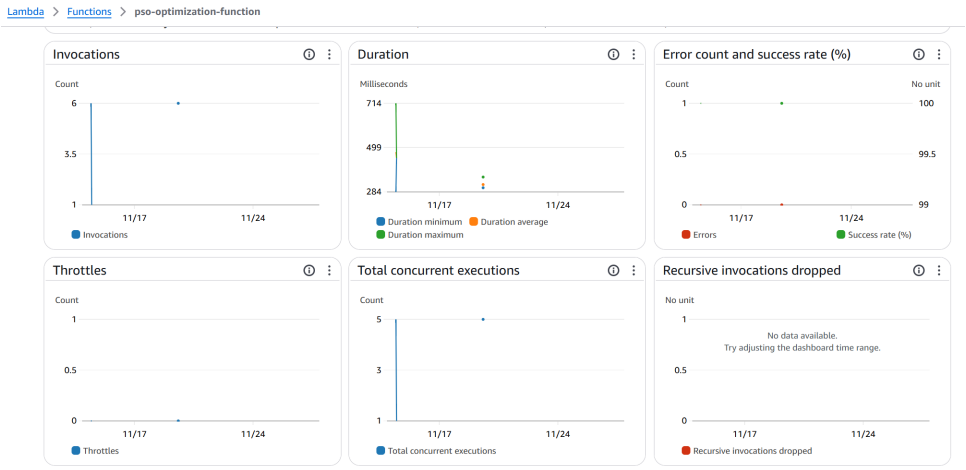


Figure 6: CloudWatch Metrics for PSO

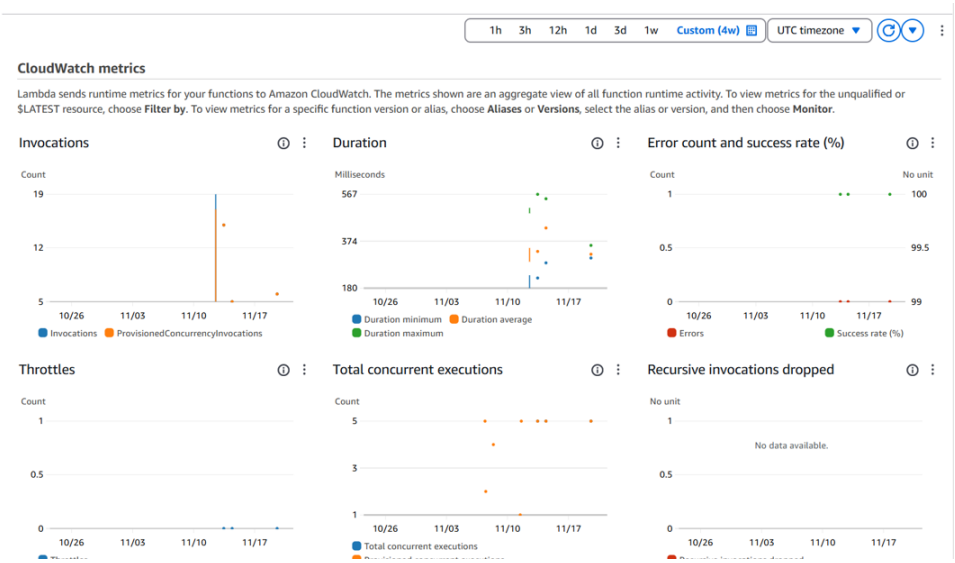


Figure 7: CloudWatch Metrics for PSO with Provisioned Concurrency.

Provisioned Concurrency : Since provisioned concurrency can gurantee a consistent performance and low latency, was enabled for the PSO during the evaluation.Figure 7, shows the cloudWatch graphs of provisioned concurrency that holds information of invocation count, total number of concurrent executions and success percentage. The performance metrics justified that 10 instances, were able to handle to concurrent invocations without any error or throttles, it maintained 100% of success rate. This proves the efficiency of the presented system especially when its applied on workloads and enterprise environments.

5 Implementation

At the implementation stage, research concludes and developed a theoretical concepts into a functional and scalable. The final outputs of this section included processed data, developed algorithms, serverless deployment configurations and its outputs at the final stage of this research.

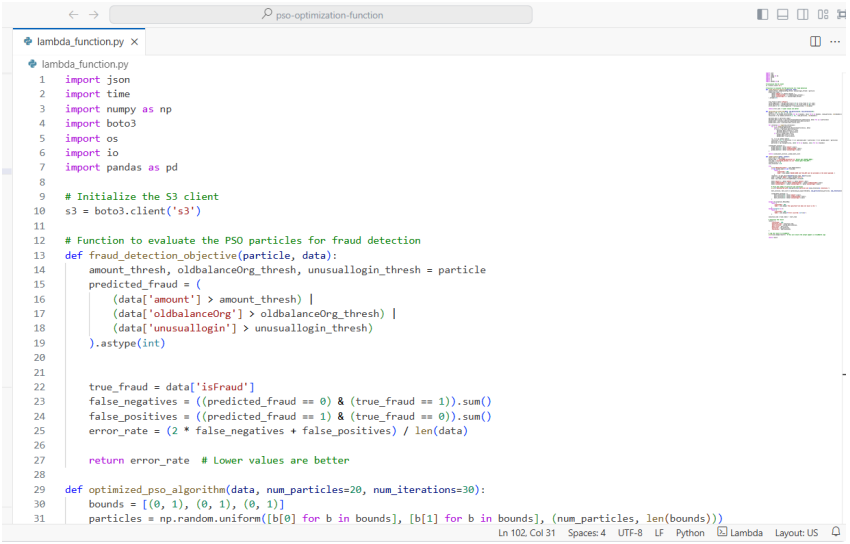


Figure 8: Code View: Deploying PSO Algorithm on Serverless AWS Lambda

Data Preprocessing and Storage: As mentioned above, the normalized transaction data was securely stored in AWS S3 bucket. This was the important step which made a seamless access for AWS Lambda functions.

Algorithms Deployment: Two different AWS Lambda functions were created for PSO and without PSO and deployed to directly with their dependencies and uploaded as zip files. The deployment was designed to provide performance under difference resource allocation to test performance variations. A figure 8, of the Lambda function code used for PSO implementation. Configure it with memory allocation variations and concurrency in order to maximize performance.

Implementing concurrent executions using VS Code : Python scripts were used on VS code to trigger concurrent invocations of Lambda functions to simulate real world situations. It showed how precise it was, to add multiple requests to the system.

Provisioned Concurrency Setup: To resolve cold start latency problems, provisioned concurrency was used. The system achieved consistent response times under high demand scenarios by creating a new alias and providing provisioned concurrency for particular lambda functions.

Failure handlings: In this implementation, also implemented robust exception handling for handling AWS S3 access errors which means the system will not halt execution and will mention the issues or errors in log.

Advanced PSO Tuning : PSO parameter adjustments that are inertia, velocity and improved globalbest convergence rates.

Execution Output: Recorded the thresholds and their error rate of both the algorithms. These performance results were logged into AWS Cloudwatch so that, it performs a detailed analysis on different configurations. Execution metrics visualization such as cost comparisons and error reductions were mentioned in the evaluation section 6.
This research provides advanced testing workflows and performance efficiency to set a benchmark for serverless deployment using optimization. This works not only focused on theoretical solutions but also focused on practical, scalable and efficient solutions.

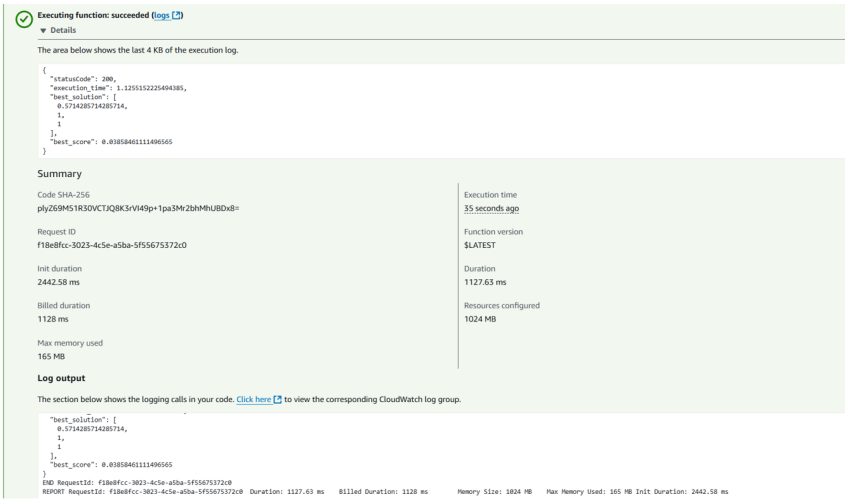


Figure 9: Evaluation Results for Without PSO at 1024 MB Allocation

6 Evaluation

In this research, the evaluation section is focused on determining the performance, efficiency and cost effectiveness of the implemented dataset with different configurations. Then compare it with and without PSO algorithms. Three case studies are addressed in this evaluation part those are: cold start performance, pre-warmed techniques and scalability of cost analysis.

6.1 Case study 1 : Performance Comparison (With and Without PSO)

1. **Without PSO** : First, tested with Grid Search approach to check the efficiency of the system. In the configuration part for resource allocation, the lambda function was configured with memory allocation of 1024 MB size and tested threshold values for amount, OldbalanceOrg and unusuallogin features.The output we got as shown in fig 9, execution time : 1.1255(s), billed duration:1128(ms),Init Duration:2442.58 (ms) ,max memory used: 165 MB. The best solution was examined through finding out the common threshold which minimize the error rate and Without PSO (Grid Search) has the best score of 0.0386 which is the error rate and weight calculation of False Negative(FN) and False Positive(FP) where the lower rte of score indicated better fraud detection.

2. **With PSO** : In next stage, Particle Swarm Optimization(PSO) algorithm is utilized to actively adapt the set thresholds used for fraud detection. The AWS Lambda configuration, memory allocation was kept same as before, 1024MB size. In PSO, a swarm of particles was initiated each of which presenting a best solution, randomly located and possessing with random velocities. These particles over multiple iterations adapted their thresholds with positions updated accordingly to personal best scores and global best solution. *Personal Best Score :* The highest score ,particles has achieved to the time. *Global Best Solution:* It is achieved by the any random particles reached its performance vector. The experiment performed

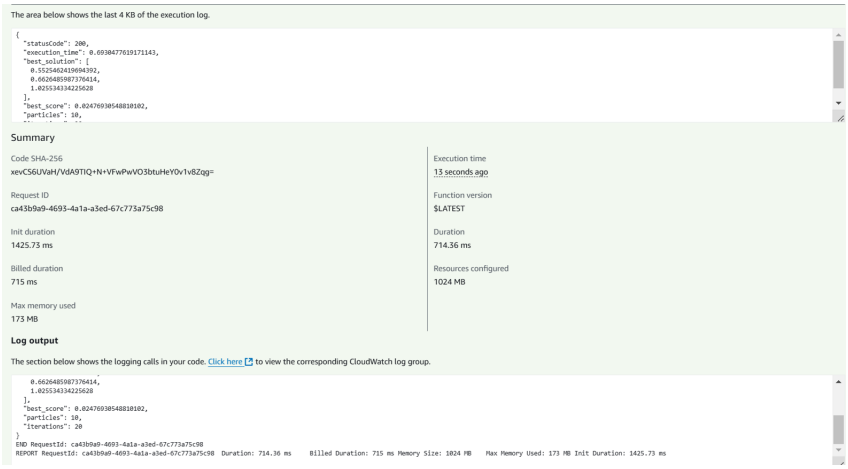


Figure 10: Evaluation Results for With PSO at 1024 MB Allocation

with 10 particles and 20 iterations to perform a better converge towards the optimal solution. As illustrated in fig 10, the execution time was improved greatly down to 0.693(s) with a billed duration:715(ms) and the max memory used:173 MB. This took less time to converge and get the optimized best score of 0.0248,which demonstrated a better improvement than the previous without PSO(Grid Search) Algorithm method for minimizing the errors.

The comparision of performance between the system without PSO and With PSO algorithm, both configured with a memory size of 1024 MB,the percentage improvement over Grid Search is shown in 2. The findings of this table shows a enhancement with the usage of PSO in relation to execution time, billed duration and accuracy. Also its consuming similarly memory size as compared to without PSO algo. The results described here are effective which showing the benefits of using the PSO for real-time fraud detection optimization.

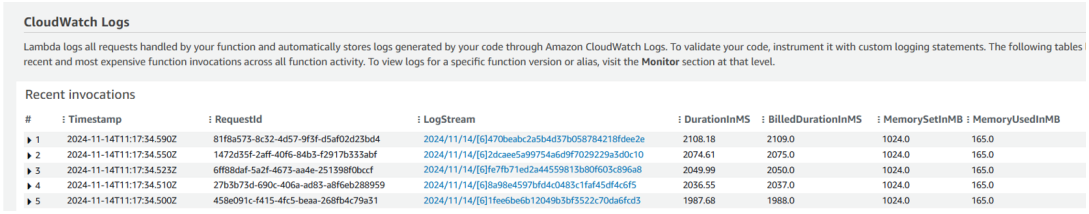
Metric	**PSO**	**Grid Search**	**Percentage Improvement**
Execution Time (s)	0.693	1.1255	38.45% Faster
Best Score	0.0248	0.0386	35.75% Better Accuracy
Billed Duration (ms)	715	1128	36.62% Reduction in Billed Duration
Max Memory Used (MB)	173	165	-4.62% Increase in Memory Usage
Init Duration (ms)	1425.73	2442.58	41.63% Faster

Table 2: Performance Comparison: PSO vs. Grid Search (Memory Size = 1024 MB)

6.2 Case study 2 : Performance Analysis of Provisioned Concurrency(With and Without PSO)

1. **Without PSO:** In this phase of the experiment, an attempt was made to use the provisioned concurrency feature with 10 pre-warmed lambda instances. The size of memory allocated with 1024 MB with 5 requests executing the script using Visual Studio code. The lambda function runs successfully, as seen in fig 11b, from the CloudWatch logs. The analysis showed that the function maintaining stability of performance metrics with every concurrent requests. The testing was done with the

help of python script in VS code where the requests were initiated in parallel with 5 and the logs proved fig 11a, that the function appropriately managed the load. The execution time it took was 1.9845(s), billed duration :1988(ms), best score :0.0383 and max memory used:165MB



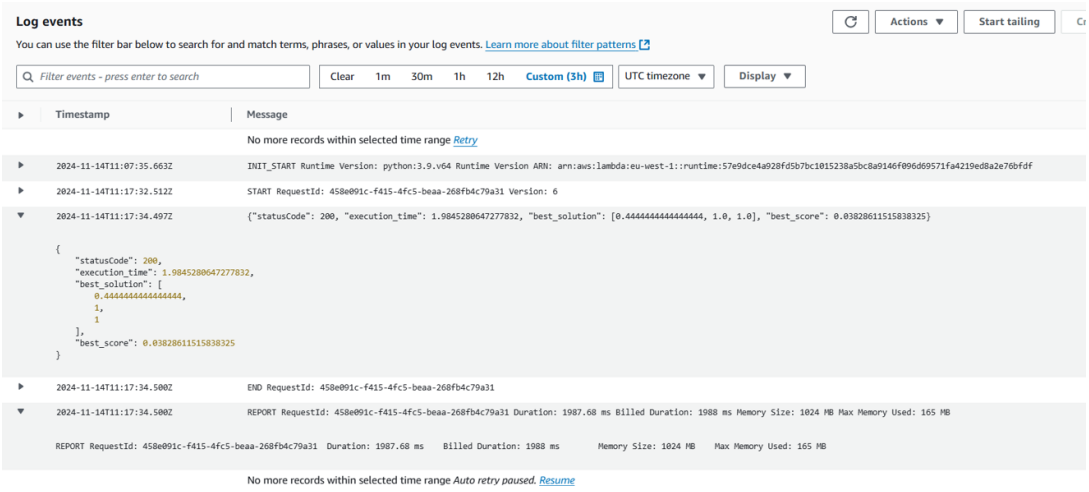
CloudWatch Logs

Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables i
recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

Recent invocations

#	Timestamp	RequestId	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
1	2024-11-14T11:17:34.590Z	81f8a573-8c32-4d57-9f3f-d5af02d23bd4	2024/11/14/[6]470beabc2a5b4d37b058784218fdee2e	2108.18	2109.0	1024.0	165.0
2	2024-11-14T11:17:34.550Z	1472d35f-2aff-40f6-84b3-f2917b333abf	2024/11/14/[6]2dcaee5a99754a6d9f7029229a3d0c10	2074.61	2075.0	1024.0	165.0
3	2024-11-14T11:17:34.523Z	6ff88daf-5a2f-4673-aa4e-251398f0bccf	2024/11/14/[6]fe7fb71ed2a44559813b80f603c896a8	2049.99	2050.0	1024.0	165.0
4	2024-11-14T11:17:34.510Z	27b3b73d-690c-406a-ad83-a8f6eb288959	2024/11/14/[6]8a98e4597bfd4c0483c1faf45df4c6f5	2036.55	2037.0	1024.0	165.0
5	2024-11-14T11:17:34.500Z	458e091c-f415-4fc5-beaa-268fb4c79a31	2024/11/14/[6]1fee6be6b12049b3bf3522c70da6fcd3	1987.68	1988.0	1024.0	165.0

(a) CloudWatch logs showing concurrent invocations without PSO



Log events

Actions ▼ Start tailing

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Filter events - press enter to search | Clear | 1m | 30m | 1h | 12h | Custom (3h) | UTC timezone ▼ | Display ▼

Timestamp	Message
	No more records within selected time range Retry
2024-11-14T11:07:35.663Z	INIT_START Runtime Version: python:3.9.v64 Runtime Version ARN: arn:aws:lambda:eu-west-1::runtime:57e9dce4a928fd5b7bc1015238a5bc8a9146f096d69571fa4219ed8a2e76bfdf
2024-11-14T11:17:32.512Z	START RequestId: 458e091c-f415-4fc5-beaa-268fb4c79a31 Version: 6
2024-11-14T11:17:34.497Z	{"statusCode": 200, "execution_time": 1.9845280647277832, "best_solution": [0.4444444444444444, 1.0, 1.0], "best_score": 0.03828611515838325}
2024-11-14T11:17:34.500Z	END RequestId: 458e091c-f415-4fc5-beaa-268fb4c79a31
2024-11-14T11:17:34.500Z	REPORT RequestId: 458e091c-f415-4fc5-beaa-268fb4c79a31 Duration: 1987.68 ms Billed Duration: 1988 ms Memory Size: 1024 MB Max Memory Used: 165 MB
	No more records within selected time range Auto retry paused. Resume


```
{
    "statusCode": 200,
    "execution_time": 1.9845280647277832,
    "best_solution": [
        0.4444444444444444,
        1,
        1
    ],
    "best_score": 0.03828611515838325
}
```



```
REPORT RequestId: 458e091c-f415-4fc5-beaa-268fb4c79a31  Duration: 1987.68 ms    Billed Duration: 1988 ms        Memory Size: 1024 MB    Max Memory Used: 165 MB
```

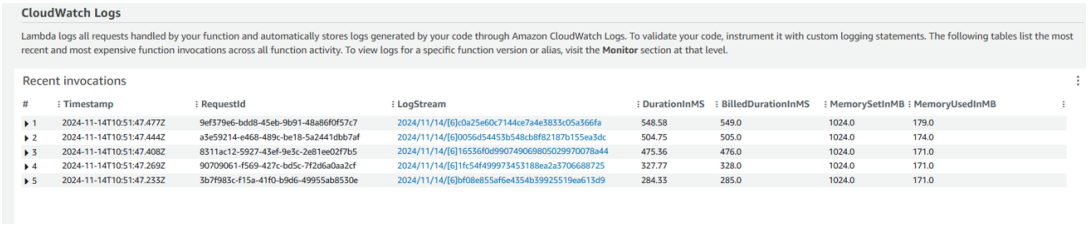

(b) Execution output of Provisioned Lambda function for without PSO

Figure 11: Performance analysis of prewarmed Lambda function without PSO.

2. **With PSO:** In the second phase of this experiment,the system was tested using Particle Swarm Optimization(PSO) algorithm, whereas the provisioned concurrency has been set as 10 instances of AWS Lambda.Assigned it with 1024 MB memory size and to test its scalability 5 simultaneous invocations were made using a python script on VS code ref in fig 12. The lambda function was able to handle all concurrent requests successfully as seen in fig 13a, cloudWatch which confirm the efficiency and resource consumption.The following observations were recorded:
Billed Duration : The billed duration was 243(ms), efficiently handling concurrent requests.
Execution time : It took around 0.22982(s) while executing the program.
Memory Utilization : The maximum memory consumed when running the program was 181 MB which is slightly above the initial measurement.
Optimization Output:With the implementation of 5 particles over 10 iterations, it was noted that the PSO algorithm gave its best result when values of [0.7772,1.0377 and 1.0167] were considered as threshold for the features: amount, oldbalanceOrg and unusuallogin respectively 13. These threshold values gave the best score of 0.0114 to which PSO is highly efficient in enhancing the low error rate in the fraud detection such as FN and FP. This results confirms the robustness of PSO in continuously varying the threshold values on a way that would effectively reduces rates in the detection of fraudulent transactions.

```python
import requests
import json
import concurrent.futures

# URL of the Lambda function
url = "https://7i22vahvrbm34jbh2bhusqdyhi0hlqcq.lambda-url.eu-west-1.on.aws/"

payload = ""
headers = {
  'Content-Type': 'application/json'
}


def invoke_lambda():
    try:
        response = requests.post(url, headers=headers, data=payload)
        return response.text
    except requests.RequestException as e:
        return f"Request failed: {e}"

with concurrent.futures.ThreadPoolExecutor() as executor:
  futures = [executor.submit(invoke_lambda) for _ in range(5)]
  for future in concurrent.futures.as_completed(futures):
    print(future.result())
```

Figure 12: Python script on VS code-With PSO



CloudWatch Logs

Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

Recent invocations

#	Timestamp	RequestId	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
1	2024-11-14T10:51:47.477Z	9ef379e6-bdd8-45eb-9b91-48a86f0f57c7	2024/11/14/[6]c0a25e60c7144ce7a4e3833c05a366fa	548.58	549.0	1024.0	179.0
2	2024-11-14T10:51:47.444Z	a3e59214-e468-489c-be18-5a2441dbb7af	2024/11/14/[6]0056d54453b548cb8f82187b155ea3dc	504.75	505.0	1024.0	174.0
3	2024-11-14T10:51:47.408Z	8311ac12-5927-43ef-9e3c-2e81ee02f7b5	2024/11/14/[6]16536f0d990749069805029970078a44	475.36	476.0	1024.0	171.0
4	2024-11-14T10:51:47.269Z	90709061-f569-427c-bd5c-7f2d6a0aa2cf	2024/11/14/[6]1fc54f499973453188ea2a3706688725	327.77	328.0	1024.0	171.0
5	2024-11-14T10:51:47.233Z	3b7f983c-f15a-41f0-b9d6-49955ab8530e	2024/11/14/[6]bf08e855af6e4354b39925519ea613d9	284.33	285.0	1024.0	171.0

(a) CloudWatch logs showing concurrent invocations with PSO (prewarmed setup).

```
2024-11-13T11:31:18.666Z  {"statusCode": 200, "execution_time": 0.22979402542114258, "best_solution": [0.7772360422247899, 1.03706912505537, 1.016724074111095], "best_sc…

{
    "statusCode": 200,
    "execution_time": 0.22979402542114258,
    "best_solution": [
        0.7772360422247899,
        1.03706912505537,
        1.016724074111095
    ],
    "best_score": 0.011413307430791646,
    "particles": 5,
    "iterations": 10
}

2024-11-13T11:31:18.679Z  END RequestId: 344370b6-d70d-4a4c-87be-be60d0ac89d8

2024-11-13T11:31:18.679Z  REPORT RequestId: 344370b6-d70d-4a4c-87be-be60d0ac89d8 Duration: 242.66 ms Billed Duration: 243 ms Memory Size: 1024 MB Max Memory Used: 181 MB

REPORT RequestId: 344370b6-d70d-4a4c-87be-be60d0ac89d8  Duration: 242.66 ms    Billed Duration: 243 ms Memory Size: 1024 MB    Max Memory Used: 181 MB
```

(b) Execution output of Pre-Warmed Lambda function with PSO.

Figure 13: Analysis of Prewarmed Lambda Function with PSO.

The percentage improvement over Grid Search is shown table 3,this makes a clear comparison that shows a significant increase in the performance in the use of PSO than Grid Search. Compared it with the execution time is speed up by 88.42% ,the accuracy is 70.24% higher and the billed duration is 87.77% reduced and the memory performance is optimized.These outcomes demonstrate that PSO can help to enhance real time processes greatly and thus can be considered one of the most effective and flexible solutions for enterprise applications.

Metric	PSO	Grid Search	Percentage Improvement
Execution Time (s)	0.22982	1.9845	88.42% Faster
Best Score	0.0114	0.0383	70.24% Better Accuracy
Billed Duration (ms)	243	1988	87.77% Reduction in Billed Duration
Max Memory Used (MB)	181	165	9.70% Increase in Memory Usage
Provisioned Concurrency	10	10	null

Table 3: Performance Comparison of Provisioned Concurrency

6.3 Case study 3 : Scaling Efficiency & Cost Analysis

In this phase, compared the performance of both the algorithms in performing cost analysis of 5 concurrent invocations as well as comparative study of cost involved in executing 1000 invocations. Thus, the objective was to analyze the effectiveness of both the approaches to the greater or huge amount of workload(data) and the expenses made some decisions. For instance, the same configuration will assume : AWS Lambda with 1024 MB memory size and 10 instances of provisioned concurrency. Calculations for costs were made based on the AWS Lambda Pricing[4] for provisioned concurrency and the invocation costs. As mentioned above, utilized AWS Lambda pricing here, which consists of Provisioned Concurrency instances as shown in table 4.

Table 4: Cost Comparison for 5 and 1000 Invocations using PSO and Without PSO

Metric	PSO Cost (USD)	Grid Search Cost (USD)
Provisioned Concurrency (1 hr)	$0.000041667	$0.000041667
Invocation Cost (5 runs)	$0.00025272	$0.00206752
Total Cost (5 executions)	$0.000294387	$0.002109187
Invocation Cost (1000 runs)	$0.050544	$0.413504
Total Cost (1000 executions)	$0.050585667	$0.413545667

Findings :

Cost for 5 Invocations: When the 5 number of invocations and the total cost of using PSO algorithm is $0.000294387,while Grid Search's cost is $0.002109187.Hence the cost reduction is approximately 86%.

Cost for 1000 invocations : The total 1000 invocations, the cost was $0.050585667 for PSO against $0.413545667 for Grid Search, marking a cost saving of 88% in terms of PSO.

Efficiency : The huge cost reduction takes place from the fact that PSO completed in 243 ms, while without PSO(Grid Search) takes 1988 ms, thereby PSO is giving better savings in less invocation cost which explains the AWS Lambda expense.

6.4 Discussion

In Delis [2022] research, found out PSO is an efficient in optimizations tasks, tested most in simulation environments which has limitation with real world scenarios. Well, this research fill the gaps by deploying PSO in a real world serverless architecture using AWS Lambda for real time scenarios. Moreover, Javed et al. [2024] worked on AWS Lambda

[4]`https://aws.amazon.com/lambda/pricing/`

and focused on performance and cost optimization but didnt explored a PSO. In this work, it fill the gaps by optimizing resource utilization as well as reducing costs in a serverless architectures with an integration of PSO algorithm.

However, this study shows how PSO is efficient in terms of optimizing serverless architecture. But, it only did testing on limited data, this raises a question about its more complex situations. It is mainly focused on aws lambda service, for instance the testing between cost and performance was not thoroughly examined on live data access.

7 Conclusion and Future Work

This work has been able to address the research challenge on how to use Particle Swarm Optimization(PSO) to set the correct fraud detection threshold in a serverless framework. From this study, it got illustrated that how modern AWS Lambda can work with top optimization methods to minimize cost and enhance performance in enterprises. For instance, the PSO approach achieved higher accuracy and optimized the utilization of resources compared it with Grid Search(without PSO algo.),indicating the method's suitability for real-time use.These experiments brought out the ideas that utilizing error rates as well as it is managing serverless resources. The study also explained how provisioned concurrency works in avoiding to ensure that serverless platforms are relevant to changing of data. Managing the concurrency and balancing cost was a challenge to integrate PSO with AWS Lambda during research.Additionally, the provisioned concurrency set up needed to achieve real time responsiveness while navigating cold start.

The Future work, will be expand the implementation to a cloud platform analysis between AWS Lambda, Azure Functions, Google cloud Functions and to verify the cost and performance between these cloud platforms. Researchers can also further enhance the adaptability and its practicality by allowing real time scenarios data which will make it more appropriate to test the real world scenarios issues.

References

M. Abbaszadeh, S. Soltani-Mohammadi, and A. N. Ahmed. Optimization of support vector machine parameters in modeling of iju deposit mineralization and alteration zones using particle swarm optimization algorithm and grid search method, 2020. URL `https://https://www.sciencedirect.com/science/article/abs/pii/S0098300422000978`.

K. G. Al-Hashedi and P. Magalingam. Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computer Science Review*, 19:1–28, 2016. URL `https://www.sciencedirect.com/science/article/pii/S1574013721000423`.

R. Busetti, N. El Ioini, H. R. Barzegar, and C. Pahl. Distributed synchronous particle swarm optimization for edge computing. In *2022 9th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 1–8. IEEE, 2022. doi: 10.1109/FiCloud54861.2022.00009. URL `https://ieeexplore.ieee.org/document/9910522`.

D. Chahal, M. Mishra, S. Palepu, and R. Singhal. Performance and cost comparison of cloud services for deep learning workload. *2021 ACM International Conference*

on Computing Frontiers (CF '21), 2021. URL https://dl.acm.org/doi/10.1145/3447545.3451184.

W.-H. Chen, S.-C. Hsu, and H.-B. Shen. Pso-based method for svm classification on skewed data sets. *Neurocomputing*, 198:136–144, 2016. doi: 10.1016/j.neucom.2016.04.020. URL https://www.sciencedirect.com/science/article/pii/S0925231216312668.

D. K. Choubey, S. Tripathi, P. Kumar, V. Shukla, and V. K. Dhandhania. Classification of diabetes by kernel based svm with pso. *Recent Advances in Computer Science and Communications*, 14(4):1242–1255, 2019. doi: 10.2174/2213275912666190716094836. URL https://www.eurekaselect.com/article/99638.

N. Delis. Scaling and distribution of particle swarm optimization algorithms on microsoft azure, 2022. URL https://bth.diva-portal.org/smash/record.jsf?pid=diva2:1751910. Accessed: 2024-07-30.

R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995. doi: 10.1109/MHS.1995.494215. URL https://ieeexplore.ieee.org/document/494215.

T. Gunarathne, B. Zhang, T.-L. Wu, and J. Qiu. Scalable parallel computing on clouds using twister4azure iterative mapreduce. *Future Generation Computer Systems*, 29(4):1035–1048, 2013. URL https://www.sciencedirect.com/science/article/pii/S0167739X12001379.

B. Huang, X. Zheng, Z. Lin, Y. Su, and W. Cheng. Short-term load forecasting based on pso-elm. *IEEE Access*, 11:12345–12356, 2023. doi: 10.1109/ACCESS.2023.10090675. URL https://ieeexplore.ieee.org/document/10090675.

H. Javed, A. N. Toosi, and M. S. Aslanpour. Serverless platforms on the edge: A performance analysis. *Monash Research*, 2024. URL https://research.monash.edu/en/publications/serverless-platforms-on-the-edge-a-performance-analysis.

D. Kumar, A. Kothiyal, R. Kumar, C. Hemantha, and R. Maranan. Random forest approach optimized by the grid search process for predicting the dropout students. In *Proceedings of the IEEE Conference on [Conference Name]*, pages 1–6. IEEE, 2023. doi: 10.1109/ConferenceID.2023.10616372. URL https://ieeexplore.ieee.org/document/10616372.

A. M. Manasrah and H. B. Ali. Workflow scheduling using hybrid ga-pso algorithm in cloud computing. *Scholarly Journals*, 2020. URL https://www.proquest.com/docview/2407630053?sourcetype=Scholarly%20Journals. Accessed: November 23, 2024.

Neha and T. Kumar. A review on pso-svm based performance measurement on different datasets. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 8(6):1–6, 2020.

A. Pradhan and S. K. Bisoy. A novel load balancing technique for cloud computing platform based on pso. *Journal of King Saud University - Computer and Information Sciences*, 34(7):3988–3995, 2022. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2020.10.016. URL https://www.sciencedirect.com/science/article/pii/S1319157820304961.

Y. Qawqzeh, M. Alharbi, A. Jaradat, and K. Abdus Sattar. A review of swarm intelligence algorithms deployment for scheduling and optimization in cloud computing environments. *PeerJ Computer Science*, 7:e696, 08 2021. doi: 10.7717/peerj-cs.696. URL https://peerj.com/articles/cs-696/.

M. Rostami, S. Forouzandeh, K. Berahmand, and M. Soltani. Integration of multi-objective pso based feature selection and node centrality for medical datasets. *Genomics*, (6), 2020. ISSN 0888-7543. doi: https://doi.org/10.1016/j.ygeno.2020.07.027. URL https://www.sciencedirect.com/science/article/pii/S088875432030224X.

H. Shafiei, A. Khonsari, and P. Mousavi. Serverless computing: A survey of opportunities, challenges, and applications. *ACM Comput. Surv.*, 54(11s), Nov. 2022. ISSN 0360-0300. URL https://doi.org/10.1145/3510611.

K. Shao, Y. Song, and B. Wang. Pga: A new hybrid pso and ga method for task scheduling with deadline constraints in distributed computing. *Mathematics*, 11(6), 2023. ISSN 2227-7390. doi: 10.3390/math11061548. URL https://www.mdpi.com/2227-7390/11/6/1548.

Y. Shi and R. Eberhart. A modified particle swarm optimizer. *ResearchGate*, 1998. URL https://www.researchgate.net/publication/3755900_A_Modified_Particle_Swarm_Optimizer.

S. Wang. A comprehensive survey of data mining-based accounting-fraud detection research. In *2010 International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 50–53, 2010. doi: 10.1109/ICICTA.2010.831. URL https://ieeexplore.ieee.org/document/5522816.