Improved Shuffled Frog Leaping Algorithm Using Memetic Reconfiguration

MAJOR PROJECT Part – I

SUBMITTED BY:

PIYUSH VASHISTHA (9915103265)

PRATAYA AMRIT (9915103267)

SPARSH MAJHE (9915103276)

UNDER THE SUPERVISION OF:

DR. SHIKHA MEHTA

DEPARTMENT OF CSE/IT

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY UNIVERSITY, NOIDA

OCTOBER 2018

ACKNOWLEDGEMENT

The success of this project required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along the completion of our project work. Whatever we have done is only due to such guidance and assistance and we could not forget to thank them.

We respect and thank our mentor Dr. Shikha Mehta for giving us an opportunity to do the project working fog computing and providing us all support and guidance which made us complete the project on time. We are extremely grateful to her for providing such a nice support and guidance though she had busy schedule managing the academics.

We owe our profound gratitude to our project mentor Dr. Shikha Mehta, who took keen interest on our project work and guided us all along by providing all the necessary information for developing a good system.

We would again heartily thank our internal project guide, Dr. Shikha Mehta, Department of Computer Science, for her guidance and suggestions during this project work.

We are thankful and fortunate enough to get constant encouragement, support and guidance from all teaching staffs of Department of Computer Science which helped us in successfully completing our project work.

Also, we would like to extend our sincere regards to all the non-teaching staff and Lab Technicians of Department of Computer Science for their timely support.

Signature of Students-

PIYUSH VASHISTHA (9915103265) .....

PRATAYA AMRIT (9915103267) .....

SPARSH MAJHE (9915103276) .....

Teacher's Signature .....

TABLE OF CONTENTS

1.INTRODUCTION

Nature inspired algorithms have become increasingly popular in the recent years, and most of these metaheuristic algorithms have been found to be very efficient. Nature inspired algorithms are the problem-solving methodologies which are used for optimisation of complex real-world scenarios. These techniques are inspired by the biological processes which are observed from the nature. There are various types of nature inspired algorithms namely genetic algorithm, memetic algorithm, evolutionary algorithm, particle swarm optimisation, etc.

Genetic algorithm is a method for solving both constraint and unconstrained optimisation problems that is based on natural selection, the process that drives biological evolution. They represent an intelligent exploitation of a random search used to solve optimisation problems. Although randomised, genetic algorithms are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of genetic algorithms are designed to stimulate processes in natural systems necessary for evolution, especially those follow the principles first laid down by Charles Darwin of "Survival of the fittest".Memetic algorithm is an extension of genetic algorithm, it uses local search techniques to reduce the likelihood of the premature convergence. It is a population-based approach whose magnitude is faster than traditional genetic algorithms for some problem domains. In this the population is initialised at random or using a heuristic. Then, each individual makes local search to improve its fitness. To form new population for new generation, higher quality individuals are selected.

In our work we are presenting a methodology to improve the performance of shuffled frog leaping algorithm for large-scale global continuous optimisation problems. It is developed for combinatorial optimization by Muzaffar Eusuff, Kevin Lansey and Fayzul Pasha in 2003.SFLA integrates the merits of both the evolutionary approach based memetic algorithm and particle swarm optimisation algorithm. SFLA is based on the evolution of memes that are carried by individuals and exchange of information globally within a population due to interaction between the individuals. The population in SFLA is composed of a set of frogs that are organized into diverse clusters, called the memeplexes. Each frog in the memeplex denotes a potential solution to a given optimization problem. Within every memeplex, each of the constituent frogs holds beliefs that are influenced by the beliefs of other frogs and cultivated through a process of memetic evolution, called the local search. Subsequent to a number of memetic evolutionary steps, the memeplexes are shuffled which leads to a global evolution. These processes of local search and shuffling continue till the pre-defined convergence criteria are not met. SFLA firstly generates an initial, random population P of frogs F_i of size *n*. After computing the fitness of the initial solutions, the entire population is sorted in the descending order of their fitness values. Subsequently, the frogs (F_i) are divided into *m* memeplexes M^1, M^2, M^3…M^m as follows:

Within each memeplex, the fitness of worst solution is improved by adjusting the fitness landscape according the local and global best solutions. At this stage, if the fitness of solution improves, it replaces the worst solution else a new solution is generated at random to replace the least fitted solution.

2. RESULTS OF LITERATURE SURVEY

2.1 Integrated Summary:

Paper I : Improved Shuffled Frog Leaping Algorithm by UsingOrthogonal Experimental Design

Authors : Vajiheh Dehdeleh, Adeleh Ebrahimi, Ali Broumand Nia (all are from Artificial Intelligence Department, Islamic Azad University, South of Tehran Branch,Tehran, Iran)

Published Date : 14-15 Dec, 2016

Summary : In the original Shuffled Frog Leaping Algorithm, the worst frog's fitness is improved by the generation of random frog from local best frog or global best frog. In the third step(i.e. shuffling and memetic evolution step), if there is no improvement then we replace the frog with randomly generated frog. So in this condition frog do not benefit from the useful information embedded with all the experience. This is why original SFLA algorithm sometimes cannot converge properly or it converges with slower speed. In this paper's algorithm they are discovering more useful information from the previous search experience. Each one of the experience may have some better values in each dimensions. So the paper is proposing an orthogonal learning (OL) strategy, which combines good dimensions of them by orthogonal experimental design (OED). The combined dimension values provide more effective guidance to the worst frog to fly towards global best area. The algorithm is evaluated in the set of 13 benchmark functions and results confirm that this strategy improves the performance of SFLA, offering faster global convergence.

The proposed algorithm provides a guidance vector which is formed from the combination of good dimension of the vectors of local or global best frog. Thus the method of extracting out the best combination using the combination of guidance vector, makes it best and different from all other proposed algorithm. If there is n dimensions then for each dimension, there are 2 choices: local best frog, global best frog. Therefore 2^d tests are needed to cover all the combinations, which is not practical especially with high dimensions. So the strategy for it that we are using is OED (Orthogonal Experiment Design). OED strategy discovers the best combination only with small number of experiments. In OL-SFLA guidance vector, frogs are guided towards the global best region by using useful information from Xg and Xb. For example: Both Xb and Xg have better values on different dimensions. Guidance vector (Xo) is formed from the combination of Xb and Xg.

$$Xo = Xb \ \Theta \ Xg$$

Θ : stands for combination.
Now the update the original SFLA, three shuffling equations by Xo (in place of Xb and Xg).

Pros: Uses the previous search experiences through designing learning strategy.
Cons: Results are not good on large scale of dimensions.

Paper II : An Improved Shuffled Frog Leaping Algorithm with Single Step Search Strategyand Interactive Learning Rule for Continuous Optimization.

Authors : Deyu Tang, Yongming Cai, Jie Zhao

Published Date : 6 June, 2014

Summary : Shuffled Frog Leaping Algorithm is a combination of meta-heuristic and memetic algorithm. Traditional SFLA tends to premature convergence, so the paper is presenting an improved SFLA with single step search strategy and interactive learning rule. Single step search improves the exploring ability of algorithm for higher dimensions. Interactive Learning rule strengthens the diversity of local memeplexes, as it helps in faster convergence. Lets first discuss about Single Step Search Strategy, it is based on multidimensional data correlation to determine the search direction, which improves the search accuracy of SFLA. Now talking about the algorithm, one frog in the population is associated with arbitrary two frogs in population learning from each other, which improves the global searching ability of SFLA. The search space of frogs is divided into several low dimensional space.

$$Qtemp = Qw(t) + rand(Qbest(t) - Qw(t))$$

$Qw(t + 1) = Qtemp$; $Qtemp.fitness <= Qw.fitness$
$Qw(t+1) = rand.Qtemp$; otherwise

Interactive Learning rule: Suppose there is a class of 50, students often learn from discussing with each other or mutual cooperation, rather than being taught by teacher every time. This process of learning of student mutually is called iterative learning. SFLA is a swarm intelligence algorithm. In each memplex the worst frog leans toward the best frog. In traditional SFLA between memplexes, the frogs were only shuffled but they do not lean towards each other. This reduces efficiency of learning, therefore we use Iterative Learning rule. In the first step of IL, we let the frogs in each memplex complete a local search. Then in second step frogs in all memplex learn from each other to exchange information then were shuffled. Two frogs Qa and Qb are selected randomly from population, then interactive learning rules is as follows:

$Qa(t + 1) = Qa(t) + rand(Qb - Qa(t))$; $Qj.fitness <= Qi.fitness$
$Qa(t + 1) = Qa(t) + rand(Qa(t) - Qb)$; otherwise

Pros: Solve high dimension low convergence problem, and interactive learning rule strengthens the diversity of local memplex.

Cons: We are taking the same configuration of frogs in memplex matrix thus less dynamic.

Paper III : A Modified Shuffled Frog Leaping Algorithm with Convergence of Update Process in Local Search.

Author : Wang Qiusheng, Yang Hao, Sun Xiaoyao (all are from School of AutomaticScience and Electrical Engineering Beihang University, Beijing, China)

Published Date : 2011

Summary : This is the paper contrasting SFLA, which is a meta-heuristic algorithm based on Swarm Intelligence (SI). To overcome the short coming of local search in SFLA, a convergence property is presented in the paper. And after applying the convergence property the results of modified SFLA were compared and enhanced significantly. In the classic SFLA, the frogs with the worst fitness get replaced with the local best frog or global best frog. And when it is done, it gets multiplied to random which is in range (0, 1), this range is determined by subjective and experience thus lacks theoretical foundation. So the final results we get do not converge towards best frog.
So to enhance the convergence for large scale population, we will have to perform following mathematical steps:

Step 1 :-

For k iteration-

$$X_w^{k+1} = X_w^k + \alpha(X_b^k - X_w^k) \quad \text{------ (3)}$$

α ----> real coefficient factor

Step 2 :-

For k+1 iteration, update process would be-

$$X_w^{k+2} = X_w^{k+1} + \alpha(X_b^{k+1} - X_w^{k+1}) \quad \text{----- (4)}$$

Step 3 :-

Subtract (3) from (4) –

$$\Delta X_w^{k+1} = \Delta X_w^k + \alpha(\Delta X_b^k - \Delta X_w^k)$$

$$\Delta X_w^{k+1} = (1- \alpha) \Delta X_w^k + \alpha.\Delta X_b^k \quad \text{------ (5)}$$

where-

$$\Delta X_w^{k+1} = X_w^{k+2} - X_w^{k+1}$$

$$\Delta X_w^k = X_w^{k+1} - X_w^k$$

$$\Delta X_b^k = X_b^{k+1} - X_b^k$$

Step 4 :-

Divide eq^{n} (5) by $\Delta X_w^{\,k}$ –

$$\frac{\Delta X_w^{k+1}}{\Delta X_w^{k}} = (1 - \alpha) + \alpha \frac{\Delta X_b^{k}}{\Delta X_w^{k}} \quad \text{------- (6)}$$

By characteristics of triangle inequality eq^{n} (6) reduced to-

$$\frac{|\Delta X_w^{k+1}|}{|\Delta X_w^{k}|} \leq |1 - \alpha| + \alpha \frac{|\Delta X_b^{k}|}{|\Delta X_w^{k}|} \quad \text{------- (7)}$$

Step 5:-

As $\frac{\Delta X_b^{k}}{\Delta X_w^{k}} \ll 1$, eq^{n} (7) is reduced to-

$$\frac{|\Delta X_w^{k+1}|}{|\Delta X_w^{k}|} \leq |1 - \alpha| \quad \text{------ (8)}$$

To guarantee the convergence of the series, eq^{n} (8) must satisfy-

$$|1 - \alpha| < 1 \quad \text{i.e.} \quad 1 < \alpha < 2$$

Thus the update method in the iteration process can now be represented as-

$$X_w^{k+1} = X_w^{k} + \alpha(X_b^{k} - X_w^{k})$$

Where α is a random no. in range (0,2)
i.e. α = rand * 2

Pros :

- Can provide convergence to large scale population.
- Expands the range of local search to avoid convergence trap.

Cons :

- Proposed technique is problem dependant and cannot be applied to wider engineering fields.

2.2 Some Relevant Current/Open Problems:

- An improved SFLA algorithm with strong searching capabilities. A single step strategy used to improve the local exploration and an interactive learning rule utilized to enforce the global searching capability.

- To improve the basic SFLA a new function has been introduced. This is a polygonal function which is also a time dependent function i.e. it is based on real time function and predictions of the travel speed. This function is built to stimulate the road conditions on the expressway networks. Integrating routing and selection of emergency vehicles, a dynamic dispatching model is built. This model considers various factors such as response time, time window, time-dependent travel speed and accident security and get the best possible strategy. This strategy changes with the new accidents i.e. it improves accordingly as the new accidents occur.An improved shuffled frog leaping algorithm (IFSLA) is proposed to solve the dynamic dispatching model. The improved shuffled frog algorithm uses a model known as probabilistic model of the distribution estimation algorithm which is used to generate the population of new frogs. It can avoid a local optimum of shuffled frog leaping algorithm.

- Most of them set the acceleration factor r to be a random number between 0 and 1, but this range is determined only by subjective and experience, so it is lack of theoretical foundation. It leads to the defective results that the convergence and computation efficiency of SFLA are not satisfied in practical applications.

- A modified SFLA with an application to multivariable PID design. A new frog leaping rule is proposed to improve the local exploration ability of the algorithm. The proposed SFLA is applied to optimally tune the PID controller gains for a class of linear time-invariant multivariable processes.

2.3 Problem Statement:

The benchmark shuffled frog algorithm has been proved useful in solving complex real-world problems. But there are still some problems which cannot be solved using the basic shuffled frog algorithm. To solve such problems this benchmark algorithm must be improved. In our work we are trying to improve the algorithm using memetic optimization. In memetic optimization, we reconfigure the position of frogs in memeplexes using different scheduling concepts.

2.4 Overview Of Proposed Solution Approach And Novelty/Benefits:

There have been many improvements in the shuffled frog leaping algorithm since it was developed in the year 2003. Most of the improvements have been formed focussing on the generation of new worst frog by using different mathematical calculations. Till now there have been no changes in the way the frogs are divided into memeplexes. Our main aim is to focus on this particular area and determine a new way in which the frogs should be arranged in the memeplexes. For this we have found three new ways in which the memes should be formed.

In the first way the frogs will be first arranged in the ascending order of their fitness value and then will be divided serially into 'n' number of memeplexes, i.e., first equal 'x' number of frogs in first memeplex, second 'x' number of frogs in second memeplex, and so on till 'n' memeplexes are completed.

In the second way also the frogs will be first arranged in the increasing order of their fitness values and then will be divided into groups with 'n' number of frogs, which will then be distributed into 'n' memeplexes with the frogs divided according to their position in the group, i.e., all the first number frogs in the first memeplex, all the second number frogs in second memeplex, an so on till 'n' memeplexes are completed.

In the third way too we will first sort the frogs in the ascending order of their fitness values, and then the frogs will be distributed in the memeplexes in random positions.

Benefits-

- It will lead to faster convergence of the algorithm.
- It will expand the range of local search and can avoid trapping into the local optimum.

2.5 GANTT Chart:

Category	Participation	From Date	To Date
Research Papers:			
Paper 1	Prataya Amrit	20 Aug ‘18	30 Aug ‘18
Paper 2	Sparsh Majhe	2 Sep ‘18	12 Sep ‘18
Paper 3	Piyush Vashistha	14 Sep ‘18	22 Sep ‘18
Synopsis	All members	5 Oct ‘18	10 Oct ‘18
Term Paper	All members	11 Oct ‘18	13 Oct ‘18
Report	All members	11 Oct ‘18	13 Oct ‘18
Implemenation:			
SFLA_Version1	Prataya Amrit	25 Aug ‘18	5 Sep ‘18
SFLA_Version2	Sparsh Majhe	8 Sep ‘18	16 Sep ‘18
SFLA_Version3	Piyush Vashistha	17 Sep ‘18	28 Sep ‘18

3. ANALYSIS, DESIGN AND MODELING

3.1 Overall Architecture With Component Description:

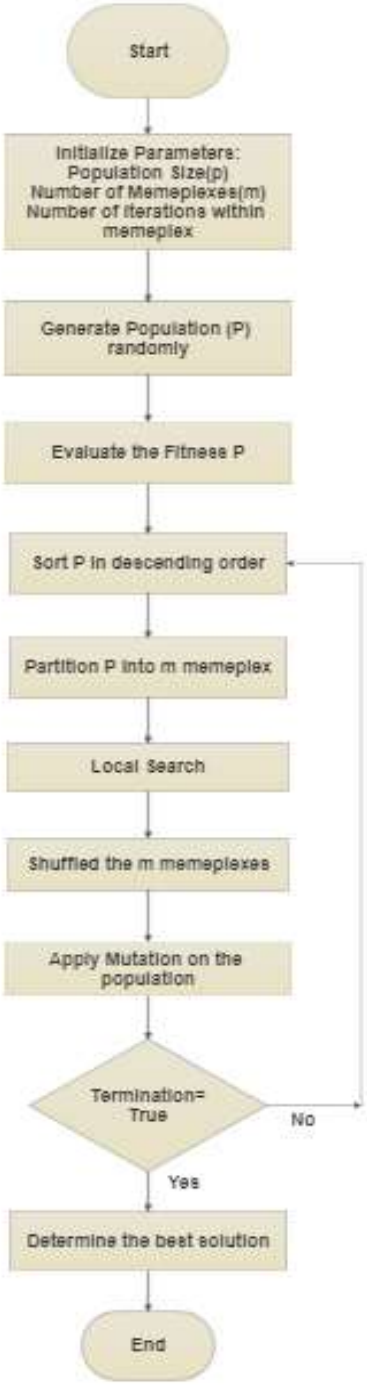

Figure: Flowchart for SFLA

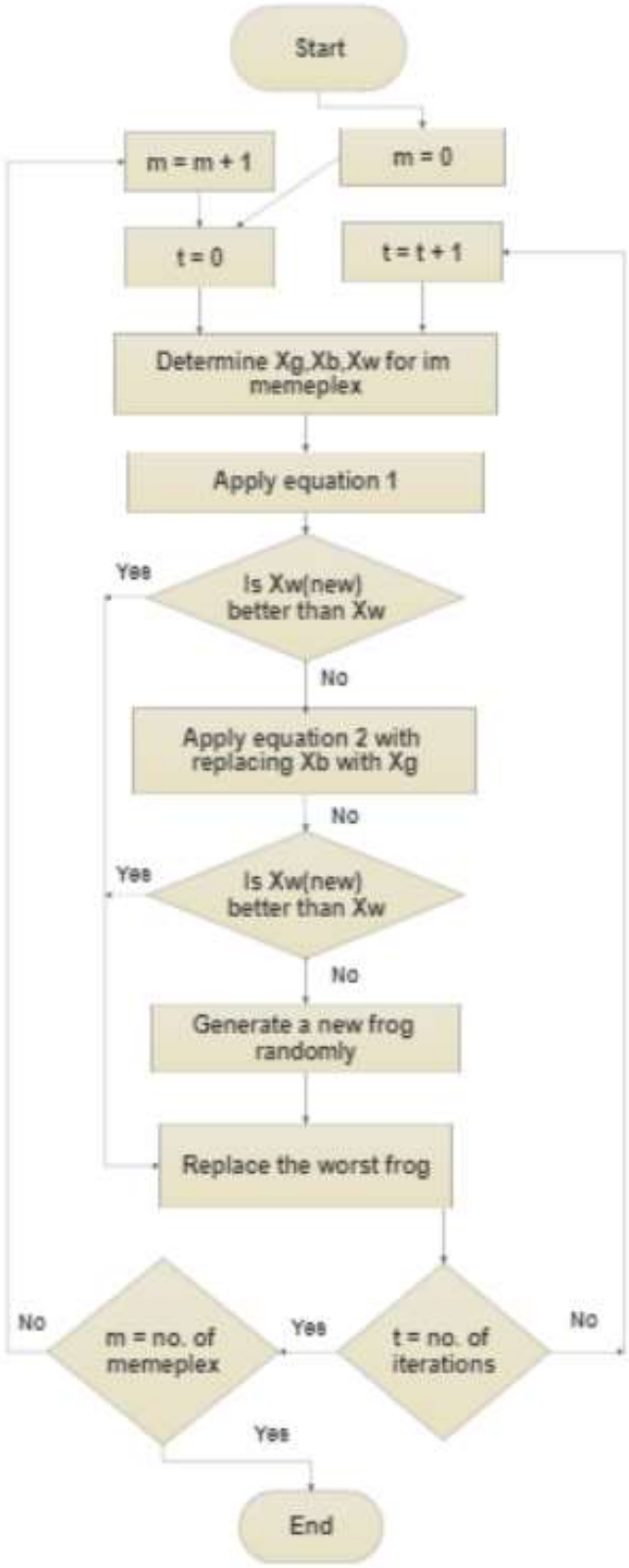

Figure: Flowchart for local search

3.2 Proposed Algorithm:

3.2.1 Original SFLA :

Starting from classic SFLA, it is a meta-heuristic approached for solving complex real world problems, it is a nature inspired cooperative for a given population. A population is as se5t of individuals. Each individual has an associated fitness value that measures how close it Is from food. SFLA consists of a set of frogs divided into certain memeplexes. The algorithm is based on the evolution of memes carried by the iterative individuals, and a global exchange of information among themselves.

Let's discuss the process through series of steps:

1. Initialize the population of the frogs of size of P.
2. Calculate the fitness of all the frogs of population and sort them in ascending order of their fitness. The fitness is the benchmark for optimization i.e the lower its value the more optimized it is. It is used to evaluate the position of frog.
3. Divide the frog population into m memeplex for each containing N frogs. Distribution would be like first frog goes to 1^{st} memeplex, second frog goes into 2^{nd} memeplex and so on.
4. We perform the local searching within the each memeplex. Let the frog with the best fitness is Xb and the frog with the worst fitness is Xw respectively. The global best frog is represented by Xg. Xw gets updated by Xg as follows:

   $$Xw' = Xw + r.(Xb-Xw)$$

   where 'r' is random number in range (0,1).
   We have to do it under certain constraints: $|Xb - Xw| < Dmax$, where Dmax is the maximum possible change of frog.
   If the above equation enhances the worst frog position towards best area then the fitness function is updated correspondingly. Else, the Xb parameter in the above equation is replaced with the global best frog (Xg) and again its fitness is not less than the worst frog fitness then, Xw is replaced by the frog which is randomly generated with arbitrary fitness.
   After completing the local search within the memeplex, all the population are shuffled and the global information are passed away those frogs in the shuffled process.
5. Local search and shuffling process continue fill the specific criteria is satisficial.

3.2.2 Improved SFLA using Memetic Reconfiguration

We have discussed various algorithms or methods which are improving the results. Some algorithms, proposed are improving the convergence rate within the memeplex. Some of the algorithms were based on the Orthogonal learning. Some have carried forward their work on memetic evolution, which leads to faster convergence by developing interactive learning methods, in which the worst frog position improved using the frogs past experiences in each dimension. One of the techniques was based on updating the local search step by assigning the value of random in the range of (0,2). This means that the

updated worst frog fitness can be better than the best frog fitness, thus avoiding the convergence trap.

Out of all these, we can observe that none of them have carried out their work on reconfiguration of memeplex. Let's now elaborate our thoughts on what do we mean by reconfiguration of memeplexes. Assuming there are (m=5) memeplexes and population size (P=25). Now let's understand the distribution of the frogs in classic SFLA algorithm with figure shown below:

F1	F2	F3	F4	F5
F6	F7	F8	F9	F10
F11	F12	F13	F14	F15
F16	F17	F18	F19	F20
F21	F22	F23	F24	F25

Figure: Classic SFLA (row wise distribution)

In our algorithm we are changing the order of frogs as shown below:

F1	F6	F11	F16	F21
F2	F7	F12	F17	F22
F3	F8	F13	F18	F23
F3	F9	F14	F19	F24
F5	F10	F15	F20	F25

Figure: ISFLA (column wise distribution)

Extending our work on further level, now we are distributing the frogs in memeplex randomly as shown in figure below:

F1	F10	F3	F18	F6
F13	F2	F9	F4	F21
F24	F15	F22	F16	F25
F17	F23	F20	F5	F14
F12	F11	F8	F19	F7

Figure: ISFLA (random distribution)

At last we would check the experimental results of the proposed algorithm with contrast to classic SFLA. We will compare worst frog, best frog, median, standard deviation and mean for the termination criteria assigned.

2.3 Risk Analysis:

Risk ID	Description of Risk	Risk Area	Probability (P)	Impact (I)	RE (P* I)	Risk Selected for Mitigation	Mitigation Plan	Contingency Plan
1	Master Node Failure	Software	M	M	M	NOY	NOY	NOY

Note - M* = Medium, NOY* = Not Done Yet

3.5 Implementation

Firstly we generate random frogs, we randomly generate population of 50 frogs. For each individual frog in the population we calculate the fitness of each frog. Fitness of each frog is calculated using the fitness function "fn". Here in this implementation we have used X^2 as the fitness function. The calculated fitness of each frog is then stored in a fitness array.
We now sort the fitness array in ascending order. Since we are dealing with the minimization problem the frog with the least fitness value will be the best frog. So our global best frog will be the frog whose fitness will be on top in the fitness array, we store this value of fitness in Xgb. Similarly the frog with the greatest fitness value will be the worst frog i.e the last frog in the fitness array.

Now we divide the population P into m memeplexes. Each memeplex represents the group of frogs. We equally divide the frogs in memeplexes. So, each memeplex has P/m frogs in it.

Here we are dividing the frogs into memeplexes in two ways.

1. Row wise distribution of frogs.
2. Column wise distribution of frogs
3. Random distribution.

We then place the frogs in the memeplex by above mentioned distribution ways. We now find the local best and local worst frogs from the memeplex. The local best is the best frog in a memeplex and the local worst frog in the worst frog in a memeplex. The best frog in a memeplex is the frog who is at the first position in the memeplex and the worst frog is the frog who is at the last position in that memeplex.

If the fitness of the newly generated frog is less than the fitness of the local worst frog, then we replace the local worst frog with the newly generated frog. This newly generated frog uses the formula : $X_w' = X_w + \text{rand}(X_b - X_w)$

If the fitness of the newly generated frog is greater than the local worst frog then again we calculate the new position of the local worst frog with respect to the global best frog.

$$X_w' = X_w + \text{rand}(X_g - X_w)$$

If the fitness of the frog with the newly generated position (w.r.t. global best frog) is less than the fitness of the local worst frog then we replace the local worst frog with the new frog generated randomly.

$$X_w' = X_w * D_{max} + X_w * -D_{max}$$

Each time we replace the local worst frog we update the fitness array i.e. we replace the fitness of the local worst frog with the fitness of the newly generated frog.

We run this algorithm 10,000 times and we iterate this whole code 25 times. Each iteration gives us one best frog i.e. which is on the top of the fitness array. We store these 25 values in another array. After this we calculate the mean, median and mode of these 25 values.

We do this for all the mentioned distribution and then compare the results i.e. the mean, median, standard deviation, best frog and worst frog for all those distributions.

Begin;
Generate random population of P solutions (frogs);
For each individual i in P: calculate fitness f (i);
Sort the population P in ascending order of their fitness;
Determine the fitness of global best frog X_{gb} as f_{gb};
Divide P into m memeplexes;
// here we change the configuration of frogs assigning in memeplexes(as discussed in theory)
For each memeplex m
Determine the fitness of local best(X_{lb})frog as f_{lb};
Determine the fitness of local worst(X_w) frogs as f_w ;
For each iteration k
For each dimension d in individual i
// Improve the worst frog fitness using Eqs. (1) with respect local best frog(X_{lb});
Change in frog position (D_d) = rand () x (X_{lb}- X_w) (1) (D_{max} >= D_d >= -D_{max})
New Position X_w = X_w + D_d ;
Compute fitness of X_w ;
// If the fitness of the new frog is less than the fitness of the local worst frog;
If fitness improves
New Position X_w = X_w + D_d ;
Else
// Improve the worst frog position using Eqs. (2) with respect global best frog(X_{gb});
Change in frog position (D_d) = rand () x (X_{gb} - X_w) (2) (D_{max} >= D_d >= -D_{max})
New Position X_w = X_w + D_d ;
Compute fitness of X_w ;
// If the fitness of the new frog is less than the fitness of the local worst frog;
If fitness improves
New Position X_w = X_w + D_d ;
// Generate the new frog and replace the local worst frog with this random frog;
Else
New Position X_w = rand() * D_{max} + rand() *(- D_{max});
End;
End;
End;
Combine the evolved memeplexes;
Sort the population P in descending order of their fitness;
Check if termination is true;
End;
End;

Pseudo code of Improve Shuffled Frog Leaping algorithm.

4.APPENDIX

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 13 10:32:21 2018

@author: PRATAYA, SPARSH, PIYUSH
"""
import numpy as np
import matplotlib.pyplot as plt
import random
import pandas

population = 50
memplex = 5
variables = 10
upperlimit = 100
lowerlimit = -100
Dmax = 100
total = 0
global_best = 0
n = int(population / memplex)
miteration = 8
bestfrog = []
worstfrog = []
fitness = [0 for i in range(population)]


# calculating the fitness value
def fitness_fn(frogs):
```

```
    global fitness
    for i in range(population):
        total = 0
        for j in range(variables):
            total = total + frogs[i][j] * frogs[i][j]
        fitness[i] = total
        return fitness
def fit(check):
    total = 0
    for j in range(variables):
            total = total + check[j] * check[j]
    return total

def run(dimensions):
    # generating random frogs
    frogs = [[0 for i in range(variables)] for j in range(population)]
    for i in range(population):
        for j in range(variables):
            frogs[i][j] = (random.random() * upperlimit) + (random.random() * lowerlimit)
            if frogs[i][j] < 0:
                frogs[i][j] = frogs[i][j] + 100
    for z in range(population):
        fitness[z] = fit(frogs[z])
    # arranging in ascending order
    for j in range(population - 1):
        for k in range(j+1,population):
            if fitness[j] > fitness[k]:
                temp = fitness[j]
                fitness[j] = fitness[k]
                fitness[k] = temp
                temp2 = frogs[j]
```

```
            frogs[j] = frogs[k]
            frogs[k] = temp2

# assigning the global best
global_best = fitness[0]

# creating the memplexes
pop = 0
memplexes = [[[0 for i in range(variables)] for j in range(n)] for k in range(memplex)]
for z in range(10000):
    for i in range(memplex):
        for j in range(n):
            for k in range(variables):
                memplexes[i][j][k] = frogs[pop][k]
            pop += 1

    for i in range(memplex):
        print ("memplex", i)
        for j in range(miteration):
            bestfrog = memplexes[i][0]
            worstfrog = memplexes[i][n-1]
            fw = 0
            fw = fit(worstfrog)
            for t in range(variables):
                di = random.random()*(bestfrog[t]-worstfrog[t])
                if(di<-Dmax):
                    di=Dmax
                if(di>Dmax):
                    di=Dmax
                worstfrog[t]=worstfrog[t]+di
            fn = 0
```

```
fn = fit(worstfrog)
print("old worst ",fw," new worst ",fn)
if(fn > fw):
   bestfrog=frogs[0]
   for u in range(variables):
      di = random.random()*(bestfrog[u]-worstfrog[u])
      if(di<-Dmax):
         di=Dmax
      if(di>Dmax):
         di=Dmax
      worstfrog[u]=worstfrog[u]+di
   fn = fit(worstfrog)
   print("old worst1 ",fw," new worst1 ",fn)
   if(fn < fw):
      memplexes[i][n-1] = worstfrog
      fitness[i*5]=fn
      for jj in range(population - 1):
         for kk in range(j+1,population):
            if fitness[jj] > fitness[kk]:
               temp = fitness[jj]
               fitness[jj] = fitness[kk]
               fitness[kk] = temp
   elif(fn > fw):
      for v in range(variables):
         worstfrog[v] = random.random() * upperlimit + random.random() * lowerlimit
      memplexes[i][n-1] = worstfrog
      fn = fit(worstfrog)
      print("old worst2 ",fw," new worst2 ",fn)
      fitness[i*5]=fn
   elif(fn < fw):
      memplexes[i][n-1] = worstfrog
```

```
                fitness[i*5]=fn
                for jj in range(population - 1):
                   for kk in range(j+1,population):
                      if fitness[jj] > fitness[kk]:
                         temp = fitness[jj]
                         fitness[jj] = fitness[kk]
                         fitness[kk] = temp
   y = fitness[0]
   print(y)


if __name__ == "__main__":
   for d in range(0, 21):
      run(d)
```

A. REFERENCES

[1] Thai-Hoang Huynh, “A Modified Shuffled Frog Leaping Algorithm for Optimal Tuning of Multvariable PID Controllers, ” University of Technology, Ho Chi Minh City, Vietnam.

[2] Wang Qiusheng, Yang Hao, Sun Xiaoyao, “A Modified Shuffled Frog Leaping Algorithm with Convergence of Update Process in Local Search,” School of Automatic Science and Electrical Engineering, Beihang University, Bejing University,Bejing,China.

[3] Deyu Tang, yongming Cai, Jie Zhao, “An Improved Shuffled Frog Leaping Algorithm with Single Step Saerch Stratergy and Interactive Learning Rule for Continuous Optimization,” College of Medical Informaton and Engineering, GuangDong Pharmceutical University, GuamgZhou, China.

[4] Vajiheh Dehdeleh, Adeleh Ebrahimi, Ali Broumand Nia, “Improved Shuffled Frog Algorithm by Using Orthogonal Experimental Design, “ Islmic Azad University, South of Tehran Branch, Tehran, Iran.

[5] Xiahong Duan, Tiaong Niu, Qi Huang, “An Improved Shuffled Frog Leaping Algorithm and It’s Application in Dynamic Emergency Vehicle Dispatching,” School of Economics and Management, North China University of Technology, Bejing,China,

[6] N. H. Awad, M. Z. Ali, P.N. Suganthan, J. J. Liang, B. Y. Qu, “Problem Definition and evaluation Criteria for the CEC 2017 Special Session and Competition On Single Objective Real-parameter Numerical Optimization,” School of EEE, Nanyang Technological University, Singapore, modified on October 15th 2016.

[7] Muzaffar Eusuff, Kevin Lansey, Fayzul Pasha, “ Shuffled Frog Algorithm: a memetic meta-heuristic for discrete optimization,” Department of Water Resource Sacramento, California, USA, 2003.