Use Particle Swarm Optimization to Optimize Data Search

By

MD Ashraful Alam

Alamgir Hossain

&

Khondaker Sajid Alam

Computer Science and Engineering

East West University

Fall 2016

Use Particle Swarm Optimization to Optimize Data Search

Submitted by:

MD Ashraful Alam
ID: 2012-3-60-005

Alamgir Hossain
ID: 2013-1-60-040

&

Khondaker Sajid Alam
ID: 2013-1-60-041

Supervised by:

Dr. Shamim H. Ripon

A project submitted in partial fulfillment for the degree of B.Sc. in Computer Science and Engineering

In the

Faculty of Science and Engineering

Department of Computer Science and Engineering

East West University

Fall 2016

Declaration

We hereby declare that, this submission is our own work and that to the best of our knowledge and belief it contains neither nor facts previously published or written by another people. Further, it does contain material or facts which to a substantial extent has been accepted for the award of any degree of a university or any our institution o territory except where an acknowledgement.

(MD Ashraful Alam)

(Alamgir Hossain)

(Khondaker Sajid Alam)

Letter of Acceptance

The project entitled **"Use of Particle Swarm Optimization to Optimize Data Search"** submitted by MD Ashraful Alam (2012-3-60-005), Khondaker Sajid Alam (2013-1-60-041) and Alamgir Hossain (2013-1-60-040), to the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh is accepted by the department in partial fulfillment of requirements for Award of the degree of Bachelor of Science in Computer Science and Engineering on December, 2016.

Board of Examiners

ABSTRACT

In computer science and engineering Particle Swarm Optimization (PSO) is a very good clustering for swarm optimization. This is very easy to implement &there are few parameters to adjust. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pBest and lBest locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pBest and lBestlocations. In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods. Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement. We actually used some of the features of basic PSO. In basic PSO velocity measure is a very important fact as well as position update. But we have made a modified version of PSO by using some of the features of the general PSO because general PSO is not so much comfortable with our dataset.

Acknowledgment

First of all thanks to ALMIGHTY ALLAH for the uncountable blessings on us. Thanks to our Supervisor **Dr. Shamim H. Ripon** for providing me this opportunity to test our skills in the best possible manner. He enlightened, encouraged and provided us with ingenuity to transform our vision into reality.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction and Motivation

Particle Swarm Optimization (PSO) was developed by Russell Eberhart and James Kennedy in 1995. Initially, these two began creating PC programming recreations of flying creatures rushing around sustenance sources, then later acknowledged how well their calculations dealt with enhancement issues. PSO is initially credited to Kennedy, Eberhart and Shi and was initially planned for reenacting social behavior, as an adapted representation of the development of creatures in a winged animal run or fish school. The calculation was improved and it was seen to perform streamlining. The book by Kennedy and Eberhart depicts numerous philosophical parts of PSO and swarm knowledge. A broad study of PSO applications is made by Poli. Recently, an extensive survey on hypothetical and test chips away at PSO has been distributed by Bonyadi and Michalewicz.

Inside the field of PC illustrations, the primary forerunners of PSO can be followed back to the work of Reeves (1983), who proposed particle or molecule frameworks to model questions that are alterable and can't be effectively spoken to by polygons or surfaces. Cases of such protests are fire, smoke, water and mists. In these frameworks, particles are free of each other and their developments are administered by an arrangement of guidelines. A few years after the fact, Reynolds (1987) utilized a molecule framework to reproduce the aggregate conduct of a run of feathered creatures. In a comparable sort of reenactment, Heppner and Grenander (1990) incorporated a perch that was appealing to the mimicked flying creatures. Both models roused the arrangement of principles that were later utilized as a part of the first molecule swarm advancement calculation.

Social brain science examine, specifically the dynamic hypothesis of social effect (Nowak, Szamrej and Latané, 1990), was another wellspring of motivation in the advancement of the principal molecule swarm enhancement calculation (Kennedy, 2006). The guidelines that administer the development of the particles in an issue's pursuit space can likewise be viewed as a model of human social conduct in which people alter their convictions and dispositions to accommodate with those of their associates (Kennedy and Eberhart 1995).

In software engineering, PSO is a computational technique that enhances an issue by iteratively attempting to enhance a competitor arrangement as to a given measure of value. It takes care of an issue by having a populace of competitor arrangements, here named particles, and moving these particles around in the hunt space as indicated by straightforward scientific formulae over the molecule's position and speed. Every molecule's development is affected by its nearby best known position, but on the other hand is guided towards the best known positions in the hunt space, which are overhauled as better positions are found by different particles. This is relied upon to move the swarm towards the best arrangements. PSO is a populace based stochastic approach for tackling nonstop and discrete improvement issues.

In PSO, straightforward programming operators, called particles, move in the pursuit space of an enhancement issue. The position of a molecule speaks to a competitor answer for the streamlining issue within reach. Every molecule scans for better positions in the inquiry space by changing its speed as per principles initially propelled by behavioral models of winged animal running. PSO has a place with the class of swarm insight strategies that are utilized to take care of streamlining issues.

PSO is a metaheuristic as it makes few or no suppositions about the issue being improved and can look expansive spaces of applicant arrangements. Be that as it may, metaheuristics, for example, PSO don't ensure an ideal arrangement is ever found. All the more particularly, PSO does not utilize the slope of the issue being upgraded, which implies PSO does not require that the streamlining issue be differentiable as is required by great enhancement strategies, for example, inclination drop and semi newton techniques.

PSO is a population based calculation. In this regard it is like the hereditary calculation. An accumulation of people called particles move in ventures all through an area. At every progression, the calculation assesses the target work at every molecule. After this assessment, the calculation settles on the new speed of every molecule. The particles move, then the calculation reconsiders.

The motivation for the calculation is groups of winged animals or bugs swarming. Every molecule is pulled in to some degree to the best area it has discovered as such, furthermore to the best area any individual from the swarm has found. After a few stages, the populace can blend around one area, or can combine around a couple of areas, or can keep on moving.

The particle swarm work endeavors to enhance utilizing a Particle Swarm Optimization Algorithm.

1.2 Why particle swarm optimization is important?

PSO offers numerous similitudes with developmental calculation procedures, for example, Genetic Algorithms (GA). The framework is instated with a populace of irregular arrangements and hunt down optima by overhauling eras. Be that as it may, dissimilar to GA, PSO has no development administrators, for example, hybrid and change. In PSO, the potential arrangements, called particles, fly through the issue space by taking after the present ideal particles.

Every molecule monitors its directions in the issue space which are connected with the best arrangement (wellness) it has accomplished as such. (The wellness esteem is additionally put away.) This esteem is called pBest. Another "best" esteem that is followed by the molecule swarm streamlining agent is the best esteem, got so far by any molecule in the neighbors of the molecule. This area is called lBest. At the point when a molecule takes all the populace as its topological neighbors, the best esteem is a worldwide best and is called gBest.

The particle swarm optimization idea comprises of, at every time step, changing the speed of (quickening) every molecule toward its pBest and lBest areas (nearby form of PSO). Speeding up is weighted by an irregular term, with independent arbitrary numbers being created for increasing speed toward pBest and lBest areas. In recent years, PSO has been effectively connected in numerous examination and application ranges. It is shown that PSO improves brings about a quicker, less expensive path contrasted and different strategies.

Another reason that PSO is appealing is that there are couple of parameters to conform. One rendition, with slight varieties, functions admirably in a wide assortment of utilizations. The particle swarm optimization has been utilized for methodologies that can be utilized over an extensive variety of uses, and in addition for particular applications concentrated on a particular necessity.

PSO may sound entangled, yet it's truly an exceptionally straightforward calculation. Over various cycles, a gathering of factors have their qualities balanced nearer to the part whose esteem is nearest to the objective at any given minute. Envision a rush of winged animals hovering over a range where they can notice a concealed wellspring of nourishment. The person who is nearest to the nourishment trills the loudest and alternate flying creatures swing around toward him. On the off chance that any of the other revolving around feathered creatures comes nearer to the objective than the primary, it twitters louder and the others veer over toward him. This fixing design proceeds until one of the winged creatures chances upon the sustenance. It's a calculation that is straightforward and simple to execute.

The calculation monitors three worldwide factors:

a) Target esteem or condition

b) Worldwide best (gBest) esteem showing which molecule's information is at present nearest to the Target

c) Ceasing esteem showing when the calculation ought to stop if the Target isn't found

Every molecule comprises of:

a) Information speaking to a conceivable arrangement

b) Velocity esteem demonstrating how much the Data can be changed

c) An individual best (pBest) esteem demonstrating the nearest the molecule's Data has ever gone to the Target

The particles' information could be anything. In the running winged animal's case over, the information would be the X, Y, Z directions of every flying creature. The individual directions of every winged creature would attempt to draw nearer to the directions of the fledgling which is nearer to the nourishment's directions (gBest). On the off chance that the information is an example or succession, then individual bits of the information would be controlled until the example coordinates the objective example.

The speed esteem is figured by far an individual's information is from the objective. The further it is, the bigger the speed esteem. In the flying creature's case, the people farthest from the nourishment would try to stay aware of the others by flying quicker toward the gBest fledgling. On the off chance that the information is an example or arrangement, the speed would portray how diverse the example is from the objective, and therefore, the amount it should be changed to coordinate the objective.

Every molecule's pBest esteem just demonstrates the nearest the information has ever gone to the objective since the calculation began.

The gBest esteem just changes when any molecule's pBest esteem comes nearer to the objective than gBest. Through every emphasis of the calculation, gBest continuously draws nearer and nearer to the objective until one of the particles achieves the objective.

It's additionally regular to see PSO calculations utilizing population topologies, or "neighborhoods", which can be littler, limited subsets of the worldwide best esteem. These areas can include at least two particles which are foreordained to act together, or subsets of the hunt space that particles happen into amid testing. The utilization of neighborhoods regularly help the calculation to abstain from stalling out in nearby minima.

General Algorithm

An essential variation of the PSO calculation works by having a population (called a swarm) of applicant arrangements (called particles). These particles are moved around in the pursuit space as per a couple of straightforward formulae. The developments of the particles are guided by their own best referred to position in the hunt space and also the whole swarm's best known position. At the point when enhanced positions are being found these will then come to direct the developments of the swarm. The procedure is rehashed and by doing as such it is trusted, however not ensured, that a palatable arrangement will in the end be found.

Formally, let f: $\mathbb{R}n \rightarrow \mathbb{R}$ be the cost work which must be minimized. The capacity takes a competitor arrangement as contention as a vector of genuine numbers and delivers a genuine number as yield which demonstrates the target work estimation of the given applicant arrangement. The slope off f is not known. The objective is to discover an answer a for which $f(a) \leq f(b)$ for all b in the inquiry space, which would mean an is the worldwide least. Amplification can be performed by considering the capacity $h = - f$.

Give S a chance to be the quantity of particles in the swarm, each having a position xi $\in \mathbb{R}n$ in the pursuit space and a speed vi $\in \mathbb{R}n$. Give pi a chance to be the best known position of molecule i and let g be the best known position of the whole swarm. An essential PSO calculation is then

```
for each particle i = 1,..., S do
   Initialize the particle's position with a uniformly distributed random vector: $xi \sim U(blo, bup)$
   Initialize the particle's best known position to its initial position: $pi \leftarrow xi$
   if $f(pi) < f(g)$ then
      update the swarm's best known position: $g \leftarrow pi$
   Initialize the particle's velocity: $vi \sim U(-|bup-blo|, |bup-blo|)$
while a termination criterion is not met do:
   for each particle i = 1, ..., S do
      for each dimension d = 1, ..., n do
         Pick random numbers: $rp, rg \sim U(0,1)$
         Update the particle's velocity: $vi,d \leftarrow \omega\ vi,d + \phi prp\ (pi,d-xi,d) + \phi grg\ (gd-xi,d)$
      Update the particle's position: $xi \leftarrow xi + vi$
      if $f(xi) < f(pi)$ then
         Update the particle's best known position: $pi \leftarrow xi$
         if $f(pi) < f(g)$ then
            Update the swarm's best known position: $g \leftarrow pi$
```

The qualities blo and bup are separately the lower and upper limits of the pursuit space. The end paradigm can be number of cycles performed, or an answer with satisfactory target work esteem is found. The parameters ω, φp, and φg are chosen by the specialist and control the conduct and adequacy of the PSO technique, see beneath.

Parameter Selection

The decision of PSO parameters can largely affect improvement execution. Selecting PSO parameters that yield great execution has hence been the subject of much research. The PSO parameters can likewise be tuned by utilizing another overlaying analyzer, an idea known as meta-advancement. Parameters have likewise been tuned for different improvement situations.

Neighborhoods and Topologies

The topology of the swarm characterizes the subset of particles which every molecule can trade information. The essential variant of the calculation utilizes the worldwide topology as the swarm correspondence structure. This topology permits all particles to speak with the various particles, in this way the entire swarm have a similar best position g from a solitary molecule. Be that as it may, this approach may lead the swarm to be caught into a nearby minimum, subsequently extraordinary topologies have been utilized to control the stream of data among particles. For example, in nearby topologies, particles just impart data to a subset of particles. This subset can be a geometrical one – for instance "the m closest particles" – or, all the more regularly, a social one, i.e. an arrangement of particles that is not relying upon any separation. In such a case, the PSO variation is said to be nearby best (versus worldwide best for the fundamental PSO).

Inner Workings

There are a few schools of thought with respect to why and how the PSO calculation can perform enhancement.

A typical conviction among specialists is that the swarm conduct shifts between exploratory conduct, that is, seeking a more extensive district of the hunt space, and exploitative conduct, that is, a privately situated pursuit in order to get more like a (conceivably nearby) ideal. This school of thought has been predominant since the beginning of PSO. This school of thought fights that the PSO calculation and its parameters must be picked in order to appropriately adjust amongst investigation and misuse to maintain a strategic distance from untimely joining to a neighborhood ideal yet still guarantee a decent rate of merging to the ideal. This conviction is the antecedent of numerous PSO variations.

1) Convergence

In connection to PSO the word joining normally alludes to two unique definitions:

a) Meeting of the arrangement of arrangements (otherwise known as, strength investigation, focalizing) in which all particles have met to a point in the hunt space, which could possibly be the ideal,

b) Joining to a nearby ideal where every single individual best p or, on the other hand, the swarm's best known position g, approaches a neighborhood ideal of the issue, paying little mind to how the swarm acts.

2) Biases

As the fundamental PSO works measurement by measurement, the arrangement point is less demanding found when it lies on a hub of the pursuit space, on a corner to corner, and much simpler on the off chance that it is spot on the center.
One approach is to change the calculation so it is no more delicate to the arrangement of coordinates. Note that some of these techniques have a higher computational many-sided quality (are in O (n^2) where n is the quantity of measurements) that make the calculation moderate for huge scale optimization.

Variants

Various variations of even a fundamental PSO calculation are conceivable. For instance, there are diverse approaches to introduce the particles and speeds (e.g. begin with zero speeds rather), how to hose the speed, just redesign pi and g after the whole swarm has been upgraded, and so forth.A progression of standard executions have been made by driving analysts.
The most recent is Standard PSO 2011 (SPSO-2011).

1) Hybridization
New and more advanced PSO variations are likewise consistently being acquainted in an endeavor with enhance streamlining execution. There are sure patterns in that examination; one is to make a half breed streamlining technique utilizing PSO joined with other optimizers, e.g., consolidated PSO with biogeography-based optimization, and the fuse of a viable learning method.

2) Alleviate premature
Another exploration pattern is to attempt and mitigate untimely meeting (that is, enhancement stagnation), e.g. by turning around or irritating the development of the PSO particles, another way to deal with manage untimely merging is the utilization of various swarms (multi-swarm enhancement). The multi-swarm approach can likewise be utilized to actualize multi-objective

optimization. Finally, there are advancements in adjusting the behavioral parameters of PSO amid optimization.

3) Simplifications

Another school of believed is that PSO ought to be rearranged however much as could reasonably be expected without weakening its execution; a general idea frequently alluded to as Occam's razor. Disentangling PSO was initially recommended by Kennedy and has been concentrated more extensively, where it gave the idea that enhancement execution was enhanced, and the parameters were simpler to tune and they performed all the more reliably crosswise over various improvement issues.

4) Multi-objective optimization

PSO has likewise been connected to multi-objective problems, in which the target work correlation considers pareto strength while moving the PSO particles and non-commanded arrangements are put away in order to rough the pareto front.

5) Binary, discrete, and combinatorial

As the PSO conditions given above work on genuine numbers, a usually utilized strategy to tackle discrete issues is to delineate discrete pursuit space to a constant area, to apply a traditional PSO, and after that to demap the outcome. Such a mapping can be extremely basic (for instance by simply utilizing adjusted qualities) or more sophisticated.

Notwithstanding, it can be noticed that the conditions of development make utilization of administrators that perform four activities:

a) Registering the distinction of two positions. The outcome is a speed (all the more accurately a relocation)

b) Increasing a speed by a numerical coefficient

c) Including two speeds

d) Applying a speed to a position

Typically a position and a speed are spoken to by n genuine numbers, and these administrators are basically - , *, +, and again +. Be that as it may, all these numerical articles can be characterized in a totally extraordinary manner, keeping in mind the end goal to adapt to double issues (or all the more for the most part discrete ones), or even combinatorial ones. One approach is to reclassify the administrators in light of sets.

1.3 Objectives

Inspired by the social behavior of bird flocking and fish schooling here is the objectives of our project.

- Our goal is to remove irrelevant data by filtering our original dataset.
- Our main goal is to optimize searching.
- Our another goal is to remove data duplicity

1.4 Outline

The report is organized as follows

In chapter 2, we will briefly describe the methodology of our thesis. We will give description of the every functions & parameters of our project.

In chapter 3, we will show our implementation part. We will describe part by part of each of the processes.

In chapter 4, we will give an overview of our result part. We will show it by comparing with the two different method those are General PSO & optimized PSO.

In chapter 5, we will give a summary of our total project & make a conclusion for our project.

Finally, in chapter 6 we will show all the important parts of our codes & functions of our projects in details which will be called appendix.

Chapter 2

Methodology

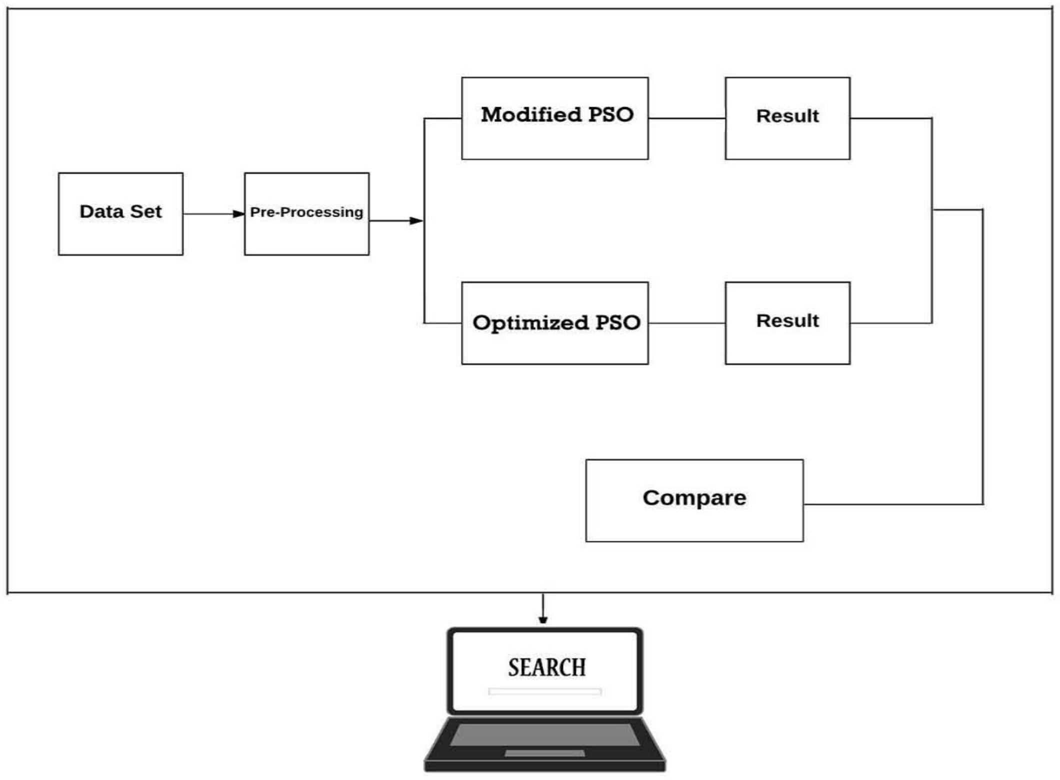


Fig 2: Methodology.

2.1 Dataset

We keep our dataset in Excel file. In excel file we keep three column which are Location, University, University website link. In our system user can search either location name or university name and user can also visit selected university website that's why kept three column. In our experiment we use 420 number of data which include ten locations thirty universities information .For a better comparison we also use 140, 25 number of data.

2.2 Pre-processing/Filtering/Ordering

Usually data may contain lots of irrelevant data and wrong information. If we can omit those irrelevant data/ wrong data then the data size will reduce and there will no wrong information that's why we develop a technique for omit irrelevant /wrong data. We apply this method to our website link column in excel file. Based on our Data we recognize a common pattern that is every website link must contain two dot (.), three slash (\) and it starts with http:\\ or https:\\. Based on this we can easily omit irrelevant link. For create a ordering by location we first need to identify unique location .When we get unique location name then we can easily create ordering by location.

2.3 Modified PSO/General PSO

Due to some limitation of PSO for our dataset we can't apply all the functionality of PSO in our dataset. That's why we have to omit some functionality of general PSO. It is very good algorithm for no ordering data. After experiment we come with a decision our PSO works better in large amount of data.

2.4 Optimized PSO

After applying our general PSO we observe that we can easily optimize our general PSO by adding a new function. For getting this optimization we must need ordering data otherwise it works same as our general PSO. When we search by location that time a location may contain lots of data. So it actually works perfect when we search by location. There is huge chance when we initialize some particle that time we can found user expected location if found then it go through to the excel file where data is in sorted by location .If not found it works as general our General PSO.

2.5 Result

We also shown the result of General PSO and Optimized PSO. What is the output if we select search by location, what is the output if we select search by university, all the output we shown in this book in result section.

2.6 Proof of Concept

To proof of concept we develop a tool. For developing this tool ,we use JavaFx. In this tool user can search by university name or location. Then the output will show in list view. On the basis of selection of university name user can also visit university website in our tool. One window pop up only one time and multiple scene are switching between them.

Chapter 3

Implementation

3.1 General PSO

The general PSO is worldwide renowned. It has a lot of variants.

The pseudo code of the procedure is as follows.

```
For each particle

{
    Initialize particle
}

Do  until  maximum  iterations  or  minimum  error
criteria
{
    For each particle
    {
        Calculate Data fitness value
        If the fitness value is better than pBest
        {
            Set pBest = current fitness value
        }
        If pBest is better than gBest
        {
            Set gBest = pBest
        }
    }

    For each particle
    {
        Calculate particle Velocity
        Use  gBest  and  Velocity  to  update  particle
Data
    }
 }
```

3.2 Flowchart

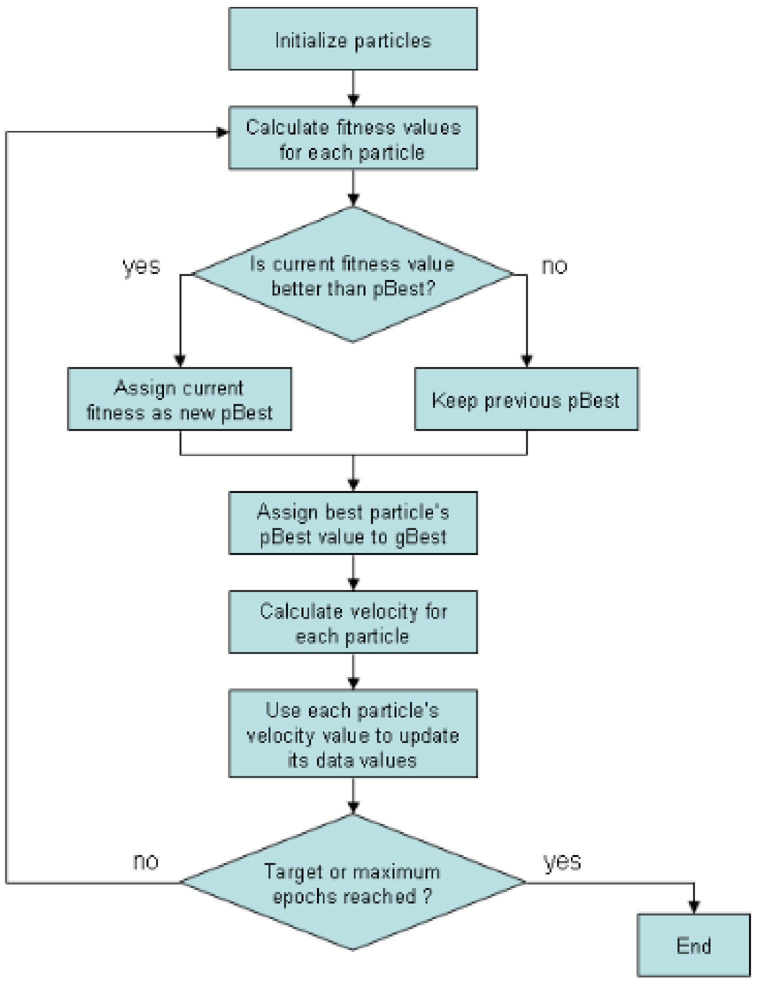

Initialize particles
Calculate fitness values for each particle
yes
Is current fitness value better than pBest?
no
Assign current fitness as new pBest
Keep previous pBest
Assign best particle's pBest value to gBest
Calculate velocity for each particle
Use each particle's velocity value to update its data values
no
Target or maximum epochs reached ?
yes
End

Figure 2. Flow diagram illustrating the particle swarm optimization algorithm.

Fig 3.2: Flowchart of PSO

At first we have to initialize all the particle by randomly generate function. Then start a 'for loop' which terminate when the maximum iteration reached. Then we have to calculate the fitness value for each particle. If current fitness value is better than this particle Best value then allocate the current best value as Best value, if not keep previous one as best . Then assign best particle best value among all particles as global best value.

Then calculate the velocity of each particle by this equation,

```
v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2
* rand() * (gbest[] - present[]) (a)
```

```
Then update the position by this equation,


present[] = persent[] + v[] (b)
```

This process continues till the Target or maximum Iteration reached.

3.3 Limitation

We gather some limitation Of Universal PSO with our data set while we trying to implement it in our data set. In our data set we keep 10 countries 10 university name and their about page URL. In our system user can search by location or search by university name. When we search by University name that time the user need to only one output, no need gather other university to the resultant particle.

We can't calculate all the particles fitness value except users search content, we don't have any parameter to calculate other particles fitness value.

We know PSO is used for problems involving global stochastic optimization of a continuous function (called the objective function). PSO can also be used for discrete optimization problems but not in string data.

As we know PSO inspired by social behavior of bird flocking or fish schooling where if a bird is near to the food then others bird will follow that bird. In Our system when we search by location there exist 10 more same locations that's mean we have multiple solution, so we can't define the exact one solution that's why we can't gather others other particle to the solution. That's why can't calculate particle velocity and update their velocity.

In our data set our data is fixed so we can't update the data by velocity. And our data position is fixed.

In PSO we have to initialize all the particle and check fitness value so that's mean we have to check out our all data, which is not optimized.

We implement basic PSO but we don't get our requirement data due to this type of limitation for our dataset that's why can't adjust with all the parameters of Basic PSO.

3.4 Basic PSO

Due to some limitation of PSO we can't use all the function of PSO which function are velocity and position update. We modified the PSO to adjust our dataset.

Pseudo code of our PSO is given below:

```
Calculate Total Row ( );
For randomly generate total particle/10,
     Initialize particle randomly ( );
     Calculate pBest ( );
END
DO
      For initialized particles,
          If  pBest fulfill its target {
                 if(SearchByUniversitySelected){
               Terminate process and show result;
                }

          }
          Calculate Global Best Test ( );
          If(GlobalBest<globalBestTest ) {
               GlobalBest = GlobalBestTest ;
               ADD result list;
          }

     END
     Update Particle ( );
While maximum iterations or minimum criteria is not
attained;
```

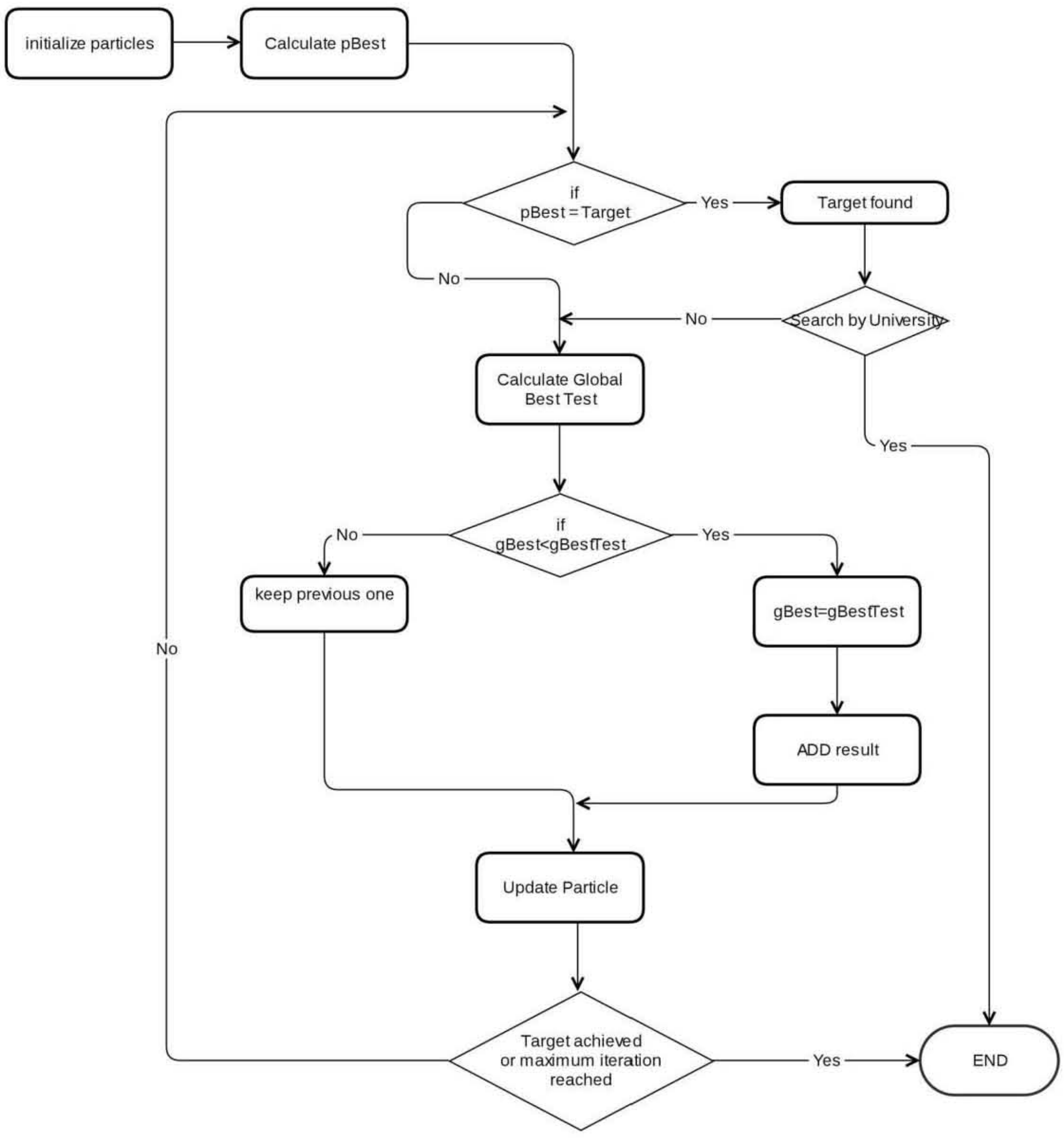


Fig 3.4: General PSO

In our PSO we at first calculate the total row of data from .xlsx file by ReadFROMXL function. Now we know the total number of data in our data set. Now we initialize randomly 10 particle from our dataset because we have 100 data. We calculate randomly 10 particle by this equation,

random_index=(maxPosition-lowest position)* new Random().nextDouble()+low;

In Initialize function we also calculate the pBest by checking if the data matches with the user input then we set pBest =3 otherwise it remain 0 . Then we start a ‘for loop’ which is terminate by fixed Iteration. Here we check that the pBest is match with the target, if match set a Boolean value as true and set the iteration as maximum iteration. Here also check if user search by university then terminate the process because when user search by university then they need only one output . If user search by location then continue the process. Then we check Global best test result by this findBestresult() where return the best particle among this 10 particles which one matches with target. Now check if our Global best result pBest is less than the global best test pBest then set the global best particle as global best test particle and add this particle to result list. This loop is continue maximum 10 times if it get target then terminate. Then update 10 more randomly generate particle. Then again start from the ‘for loop’. It continue till the maximum iteration or target found.

3.5 Pre-Processing /Filtering the URL

Usually our data has no ordering and contain irrelevant data. We can omit irrelevant data and ordering by location in this technique. So that we can optimize our PSO by extended some function.

Pseudo code of Filtering technique is given below:

```
calculateTotalRow( );
For each Row {
         Particle=new particle( );
If  (  particle.Data[2].contains  (http://www.”  || “http:” || “https:www.” ||
“https://”$$num of Dot>=2 & $numofslash>=3) {
     addFreshParticleList( );
      }
}
Calculate unique (Countryname( ) );
For each Unique Country name {
 For each Fresh Particle {
    If (Unique Country = = Fresh Particle) {
         Add result list ( );
           }
    }
}
```

3.6 Flowchart

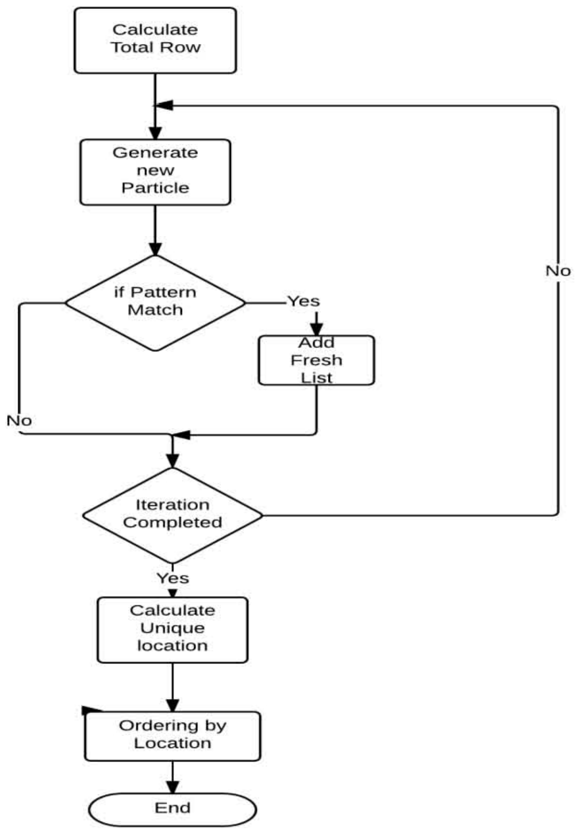


Fig 3.6: Flow chart of filtering/ordering.

We first calculate the total number of data from xlsl file. Then start a for which terminating condition is total number of particle. We check the URL does match with the url pattern? If match then we add it to fresh list . We recognize a common pattern for our data URL. Since we kept the about page link of university website so our pattern must start with http:\\www. Or http:\\ or https:\\ or https:\\ and it must contain at least two dot(.) and three slash(\) in our data URL. After filtering we have to ordering by country name that's why we need to determine the unique location name .We can easily calculate unique location by a for loop. After determining the unique location name then we can easily doing ordering by two for loop. Where first for loop terminate when it reached the number of total unique location and the second one terminate when it reached to the total number of fresh particle.

By this technique our data set size is reduced that's why PSO works more faster.

3.7 Our Optimized PSO

This PSO will better when user will search by location but that time our data must have to be sorted & ordered.

Here is the pseudo code as follows:

```
Calculate Total Row ( );
For total particle/10;
     Initialize Particle Randomly ( );
Check (pBest = = Target){
        ADD some location data from xlsFile ( );
       Stop loop;
DO
   For initialized particles,
      If pBest fulfill its target {
                  if(SearchByUniversitySelected){
              Terminate process and show result;
          }


           }
          Calculate Global Best Test ( );
         If (GlobalBest<globalBestTest) {
     GlobalBest = GlobalBestTest ;
                ADD result list;
          }
END
Update Particle ( );
While maximum iterations or minimum criteria is not attained;
```

Flowchart

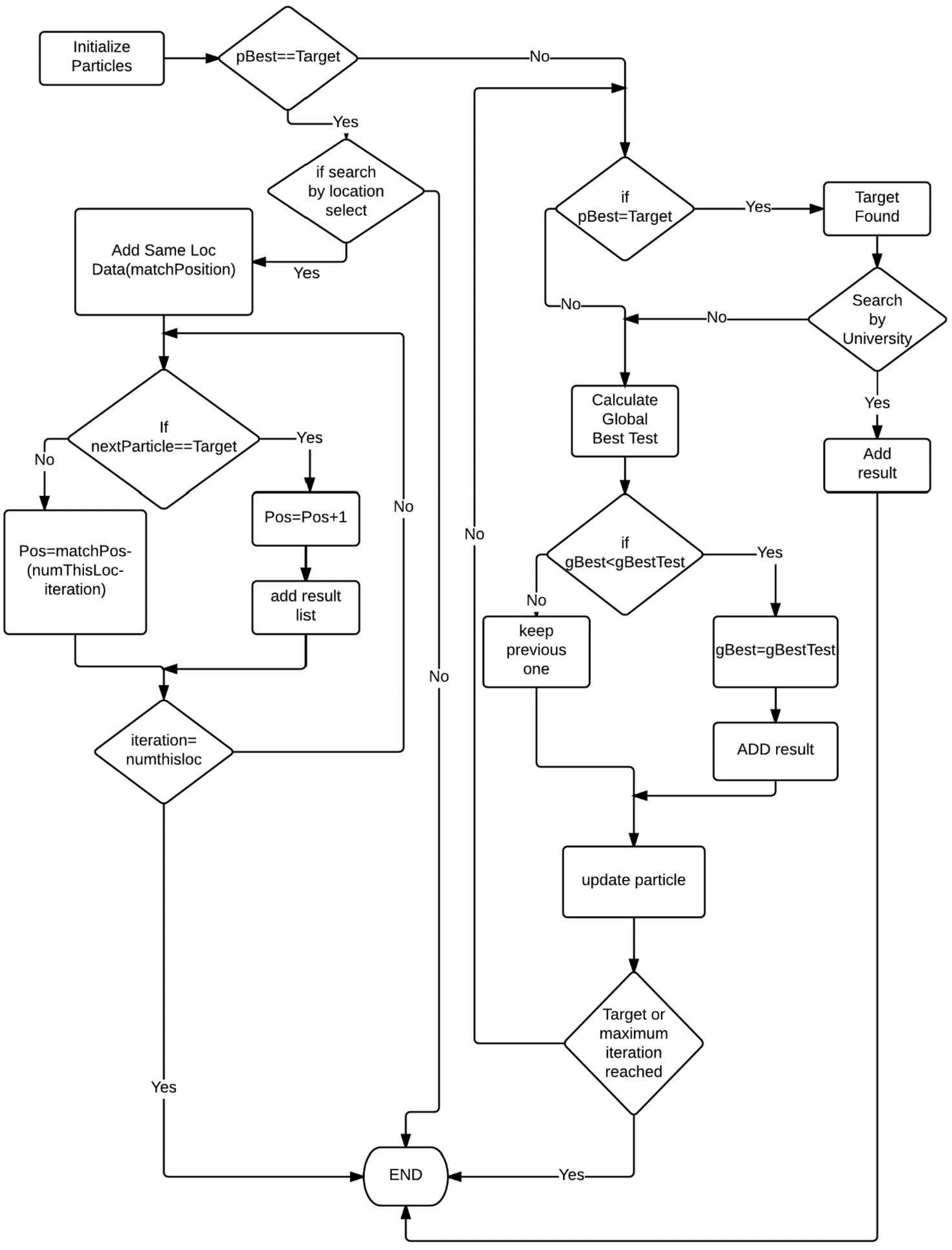


Fig 3.7: Optimized PSO.

Since we determining the pBest value when we initializing the 10 particle by randomly. When user select search by location there is a huge probability to match location name during initializing because we have 10 countries University website. If that time match we don't need to go into algorithm because our data kept into ordering by location, that time we go to our xlsl file and check the next position location isn't match? If match then it check next position and go on, if not we check into the upper side of matching index data by this equation,

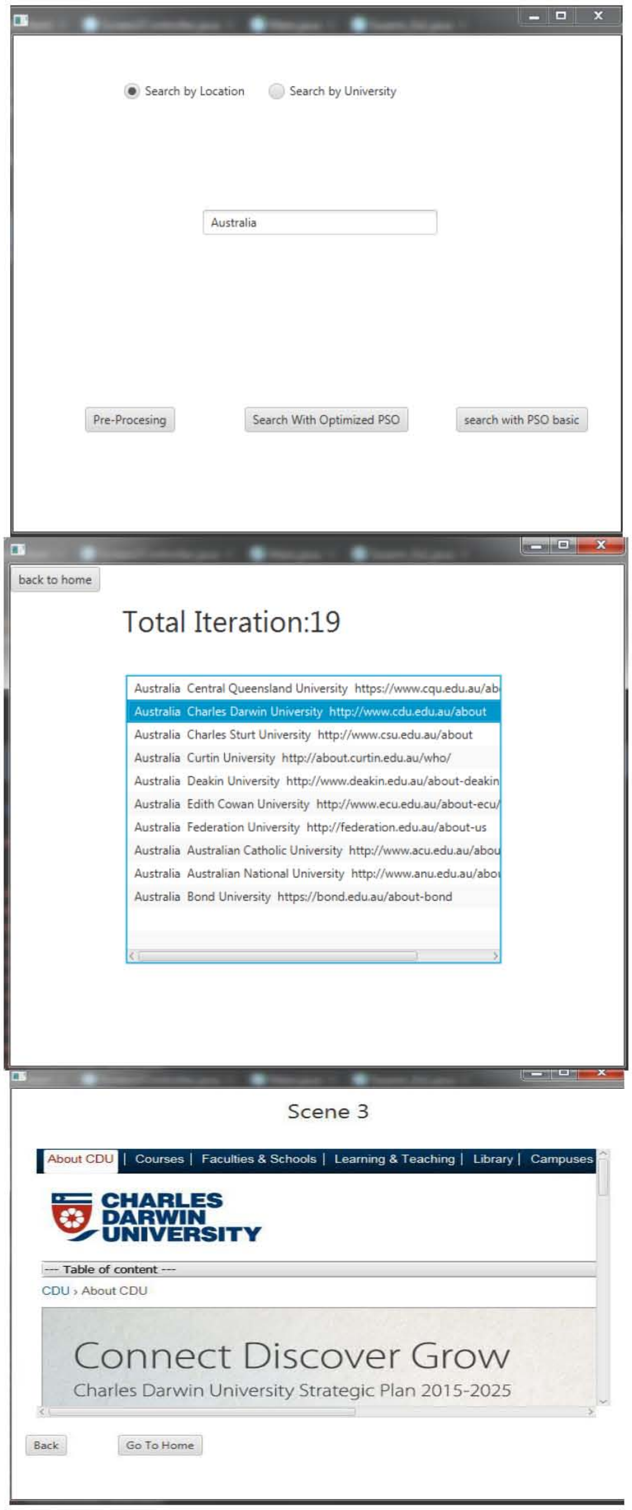
index=match_index-(totalnumofsameloc data-iteration).

This loop is continue till the iteration reached to the total number of that location data. In our case it is 10. Here we need maximum 20 iteration to get 10 location, which is totally optimized. Same as when user search by university then it follows same method. Now if during initialization user search is not found then it works as our normal PSO like,

We start a ‘for loop’ which is terminate by fixed Iteration. Here we check that the pBest is match with the target, if match set a boolean value as true and set the iteration as maximum iteration. Here also check if user search by university then terminate the process because when user search by university then they need only one output . If user search by location then continue the process. Then we check Global best test result by this findBestresult() where return the best particle among this 10 particles which one matches with target. Now check if our Global best result pBest is less than the global best test pBest then set the global best particle as global best test particle and add this particle to result list. This loop is continue maximum 10 times if it get target then terminate. Then update 10 more randomly generate particle. Then again start from the ‘for loop’. It continue till the maximum iteration or target found.

3.8 Proof of Concept

We use javaFx to build our tool User interface. It has three scene in one window. In running time it switches scene into one window. That's why it is very user friendly ,there will pop up only one window whole running time but scene will change multiple time as user requirement . We use two radio button, one Text Field, and three button. When user click on a website link then it shows the university website also.



```
Australia  Central Queensland University  https://www.cqu.edu.au/ab
Australia  Charles Darwin University  http://www.cdu.edu.au/about
Australia  Charles Sturt University  http://www.csu.edu.au/about
Australia  Curtin University  http://about.curtin.edu.au/who/
Australia  Deakin University  http://www.deakin.edu.au/about-deakin
Australia  Edith Cowan University  http://www.ecu.edu.au/about-ecu/
Australia  Federation University  http://federation.edu.au/about-us
Australia  Australian Catholic University  http://www.acu.edu.au/abou
Australia  Australian National University  http://www.anu.edu.au/abou
Australia  Bond University  https://bond.edu.au/about-bond
```

Fig 3.8: Tool View Sample.

Chapter 4

Result & Analysis

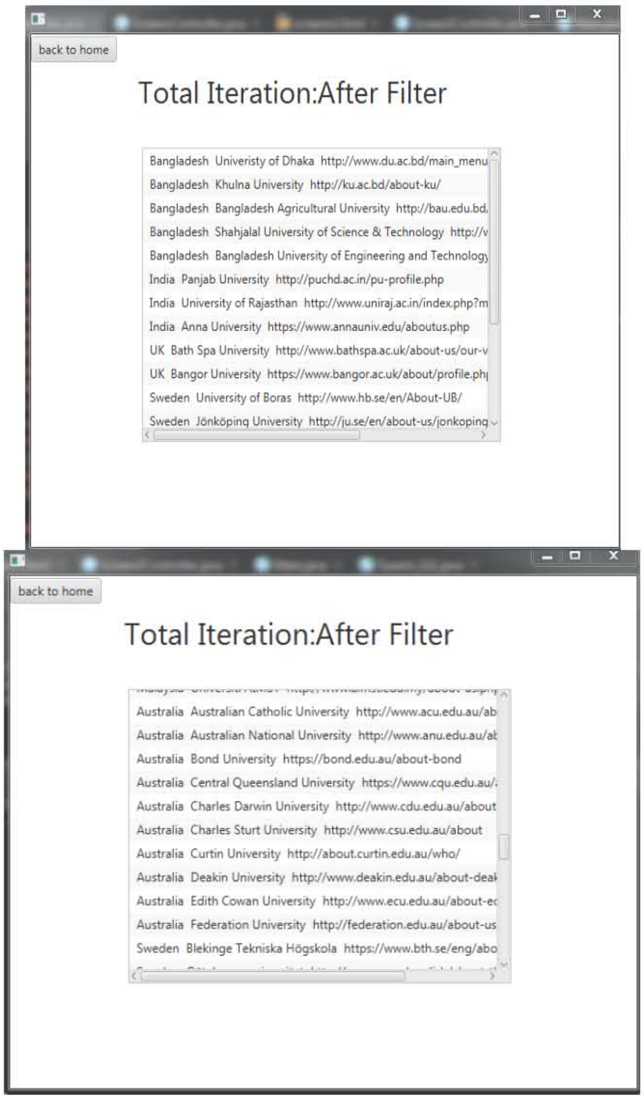
Table 4: Sample Dataset.

Country	University	About Page Link
Bangladesh	University of Dhaka	http://www.du.ac.bd/main_menu/the_university/about
Bangladesh	University	Cse105
UK	Bath Spa University	http://www.bathspa.ac.uk/about-us/our-vision-values-history
India	Panjab University	http://puchd.ac.in/pu-profile.php
Bangladesh	Khulna University	http://ku.ac.bd/about-ku/
UK	Aberystwyth University	www.cse442
UK	Faridpur university	Cse.com
Sweden	University of Boras	http://www.hb.se/en/About-UB/
India	University of Rajasthan	http://www.uniraj.ac.in/index.php?mid=1101
Malaysia	Universiti AIMST	http://www.aimst.edu.my/about-us.php
Germany	KatholischeUniversitätEichstätt	http://www.ku.de/en/ku-at-a-glance/
India	University of Ragastan	https://www.youtube
Germany	Medical University of Luebeck	https://www.uni-luebeck.de/en/university/the-university/profile.html
India	SRM University	http://srmuniversity.org/about-us/
Bangladesh	Bangladesh Agricultural University	http://bau.edu.bd/pages/view_1/MQ==
India	Anna University	https://www.annauniv.edu/aboutus.php
Malaysia	Perdana University	http:/www.knowledge.com
Malaysia	Perdana University	http://perdanauniversity.edu.my/vision-mission/
UK	Bangor University	https://www.bangor.ac.uk/about/profile.php.en
Sweden	Jönköping University	http://ju.se/en/about-us/jonkoping-university.html
Sweden	Karlstads Universitet	https://www.kau.se/en/about-university
Bangladesh	Shahjalal University of Science & Technology	http://www.sust.edu/about
Bangladesh	Bangladesh University of Engineering and Technology	http://www.buet.ac.bd/?page_id=2

Though we work with 423 data but here we show with only 23 data as sample. Our PSO works better for 423 data than this 23 data .That's mean PSO works better for large number of data. In 423 data include 10 countries 30 universities information and 120 irrelevant/wrong website link .Here the data size is 23 which including 5 irrelevant link/wrong link. We show both 23 and 423 data set output here.

4.1 Pre-processing /Filtering/ Ordering

In our system we filter data by omit the wrong URL. And make ordering by location. After apply Pre-processing technique to given data then the data size will be 18 by reducing the wrong URL and it will ordering by location. The outputs are both for 18 and 423 dataset.



Total Iteration:After Filter

```
Bangladesh  Univeristy of Dhaka  http://www.du.ac.bd/main_menu
Bangladesh  Khulna University  http://ku.ac.bd/about-ku/
Bangladesh  Bangladesh Agricultural University  http://bau.edu.bd.
Bangladesh  Shahjalal University of Science & Technology  http://v
Bangladesh  Bangladesh University of Engineering and Technology
India  Panjab University  http://puchd.ac.in/pu-profile.php
India  University of Rajasthan  http://www.uniraj.ac.in/index.php?m
India  Anna University  https://www.annauniv.edu/aboutus.php
UK  Bath Spa University  http://www.bathspa.ac.uk/about-us/our-v
UK  Bangor University  https://www.bangor.ac.uk/about/profile.ph
Sweden  University of Boras  http://www.hb.se/en/About-UB/
Sweden  Jönköping University  http://ju.se/en/about-us/jonkoping
```

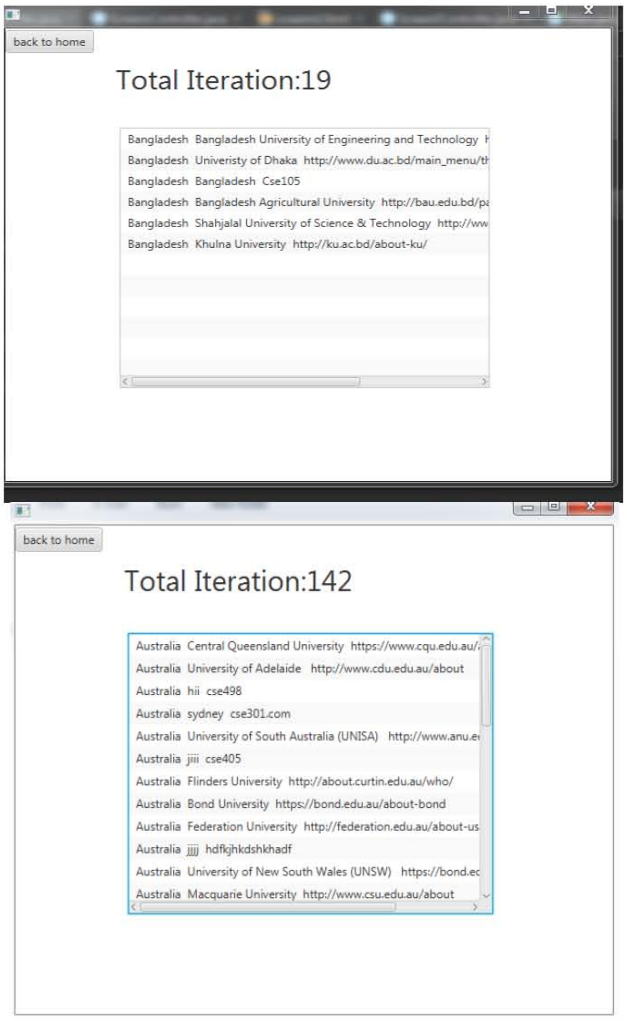
Total Iteration:After Filter

```
Australia  Australian Catholic University  http://www.acu.edu.au/ab
Australia  Australian National University  http://www.anu.edu.au/ab
Australia  Bond University  https://bond.edu.au/about-bond
Australia  Central Queensland University  https://www.cqu.edu.au/
Australia  Charles Darwin University  http://www.cdu.edu.au/about
Australia  Charles Sturt University  http://www.csu.edu.au/about
Australia  Curtin University  http://about.curtin.edu.au/who/
Australia  Deakin University  http://www.deakin.edu.au/about-deak
Australia  Edith Cowan University  http://www.ecu.edu.au/about-ec
Australia  Federation University  http://federation.edu.au/about-us
Sweden  Blekinge Tekniska Högskola  https://www.bth.se/eng/abo
```

Fig 4.1: Output of Pre-Processing.

4.2 Before Pre-Processing & Our PSO

If we select search by location and search with Location Bangladesh then the output comes with the 6 Bangladesh university including wrong link data in 19 iteration. In 423 data set search by location, it took 142 iteration for the output. This iteration number is not fixed because our algorithm worked by random number generation. Some time it could be increased or decreased. Here are the screenshot,

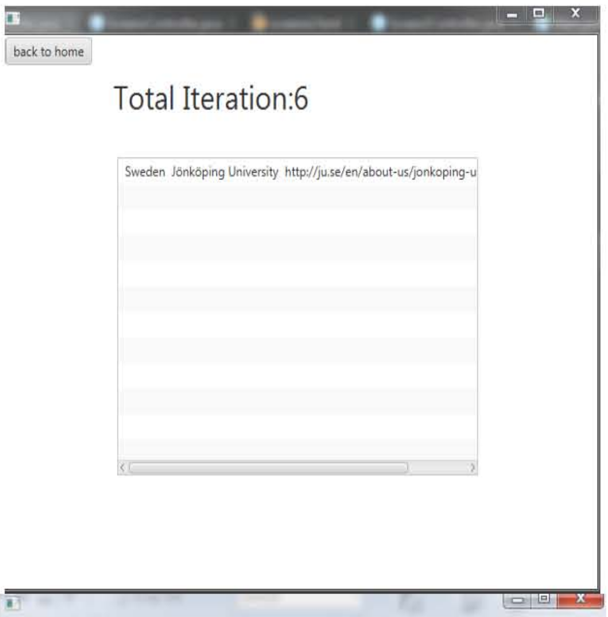


```
Total Iteration:19

Bangladesh  Bangladesh University of Engineering and Technology  h
Bangladesh  Univeristy of Dhaka  http://www.du.ac.bd/main_menu/th
Bangladesh  Bangladesh  Cse105
Bangladesh  Bangladesh Agricultural University  http://bau.edu.bd/pa
Bangladesh  Shahjalal University of Science & Technology  http://ww
Bangladesh  Khulna University  http://ku.ac.bd/about-ku/
```

```
Total Iteration:142

Australia  Central Queensland University  https://www.cqu.edu.au/
Australia  University of Adelaide   http://www.cdu.edu.au/about
Australia  hii  cse498
Australia  sydney  cse301.com
Australia  University of South Australia (UNISA)   http://www.anu.ed
Australia  jiii  cse405
Australia  Flinders University  http://about.curtin.edu.au/who/
Australia  Bond University  https://bond.edu.au/about-bond
Australia  Federation University  http://federation.edu.au/about-us
Australia  jjjj  hdfkjhkdshkhadf
Australia  University of New South Wales (UNSW)   https://bond.ed
Australia  Macquarie University  http://www.csu.edu.au/about
```

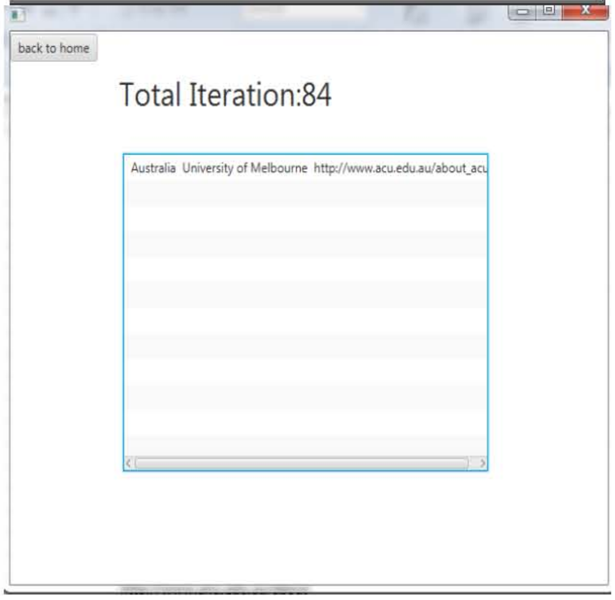
Fig 4.2.1: Search by location output of before pre-processing and our Modified PSO.

If we select search by university and search with Jönköping University of Sweden then it come with the output in 6 iteration. In 423 data set when we select search by university then it took 84 iteration for output .Again I am saying that iteration number could be increased so much or decrease because our main system is based on random number generation. Screenshot are:


back to home

Total Iteration:6

```
Sweden  Jönköping University  http://ju.se/en/about-us/jonkoping-u
```


back to home

Total Iteration:84

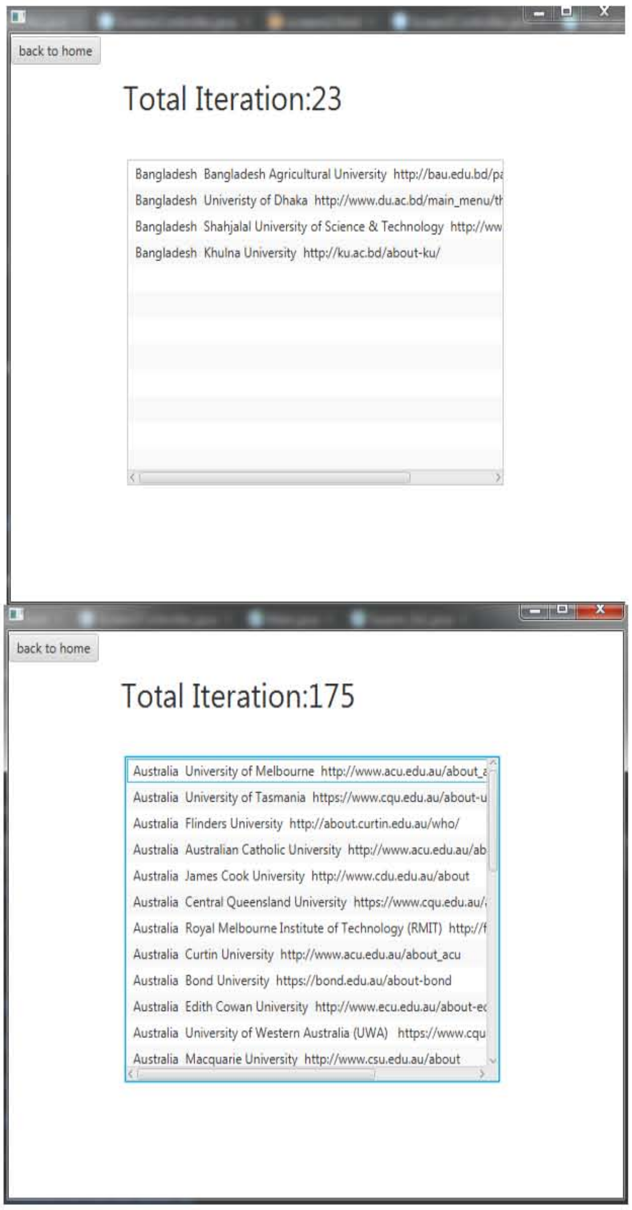
```
Australia  University of Melbourne  http://www.acu.edu.au/about_acu
```

Fig 4.2.2: Search by University name,before pre-processing and our modified PSO.

4.3 After preprocessing and Our PSO

The main facilities of pre-processing technique is it reduced the size of the data. And we won't get irrelevant data anymore. Except this facilities pre-processing is not help in normal PSO.

If we search for Bangladesh in filtering data then it comes with 4 fresh Bangladesh data in 23 iteration and our filter data has 4 Bangladesh data. In 423 dataset, when we select search by location then it took 175 iteration for bringing 30 Australian universities information and there is no wrong /irrelevant data.



Total Iteration:23

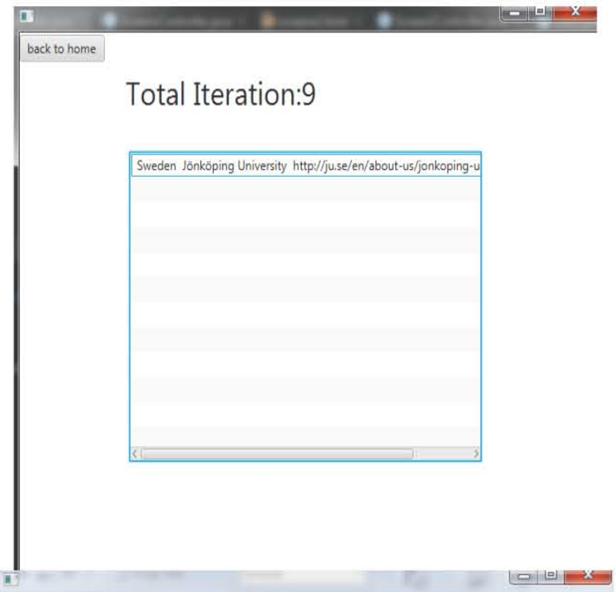
```
Bangladesh  Bangladesh Agricultural University  http://bau.edu.bd/pa
Bangladesh  Univeristy of Dhaka  http://www.du.ac.bd/main_menu/th
Bangladesh  Shahjalal University of Science & Technology  http://ww
Bangladesh  Khulna University  http://ku.ac.bd/about-ku/
```

Total Iteration:175

```
Australia  University of Melbourne  http://www.acu.edu.au/about_a
Australia  University of Tasmania  https://www.cqu.edu.au/about-u
Australia  Flinders University  http://about.curtin.edu.au/who/
Australia  Australian Catholic University  http://www.acu.edu.au/ab
Australia  James Cook University  http://www.cdu.edu.au/about
Australia  Central Queensland University  https://www.cqu.edu.au/
Australia  Royal Melbourne Institute of Technology (RMIT)  http://f
Australia  Curtin University  http://www.acu.edu.au/about_acu
Australia  Bond University  https://bond.edu.au/about-bond
Australia  Edith Cowan University  http://www.ecu.edu.au/about-ec
Australia  University of Western Australia (UWA)  https://www.cqu
Australia  Macquarie University  http://www.csu.edu.au/about
```

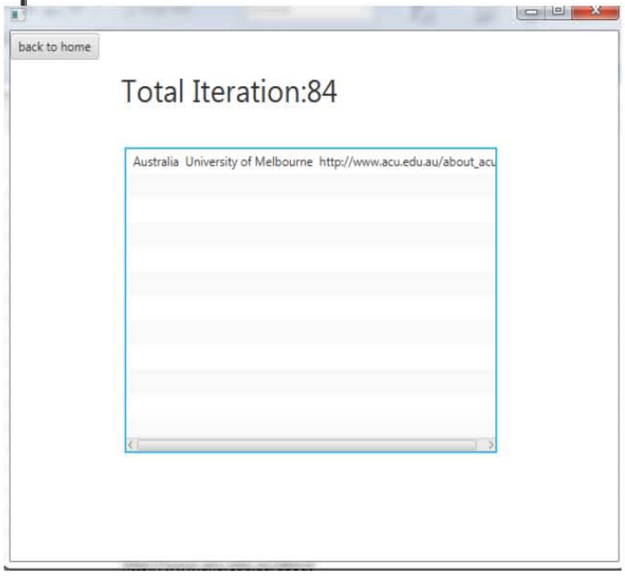
Fig 4.3.1: Search by location output, after pre-processing and our modified PSO

If we select search by university then search with Jönköping University then it comes without put in 9 iteration. In 423 dataset when we select search by university of Melbourne then the system came out with the output in 84 iteration. Here are the screenshot:


back to home

Total Iteration:9

Sweden Jönköping University http://ju.se/en/about-us/jonkoping-u


back to home

Total Iteration:84

Australia University of Melbourne http://www.acu.edu.au/about_acu

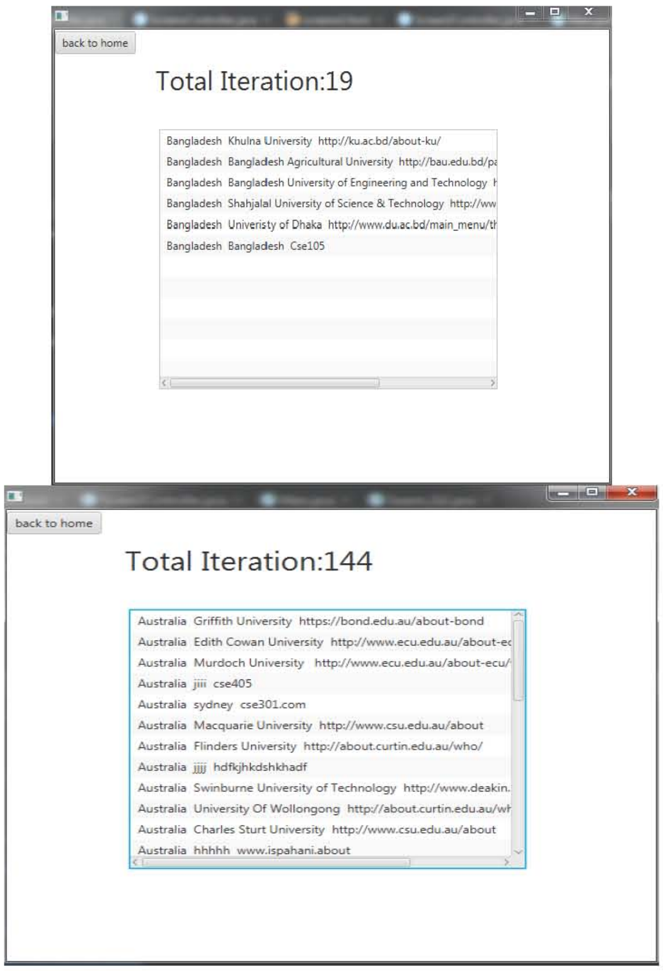
Fig 4.3.2: Search by university, after pre-processing and our modified PSO.

4.4 Before pre-processing and our optimized PSO

In this algorithm it don't works perfectly in before pre-processing data. It works same as normal PSO.

If we select search by location and search with Bangladesh then it comes with the output in 19 iteration. In 423 data set ,when we search by location then it just work as normal PSO in before pre-processing, The output which takes 144 iteration for gather 30 Australian university information including irrelevant data,

Here is the output screenshot,



Total Iteration:19

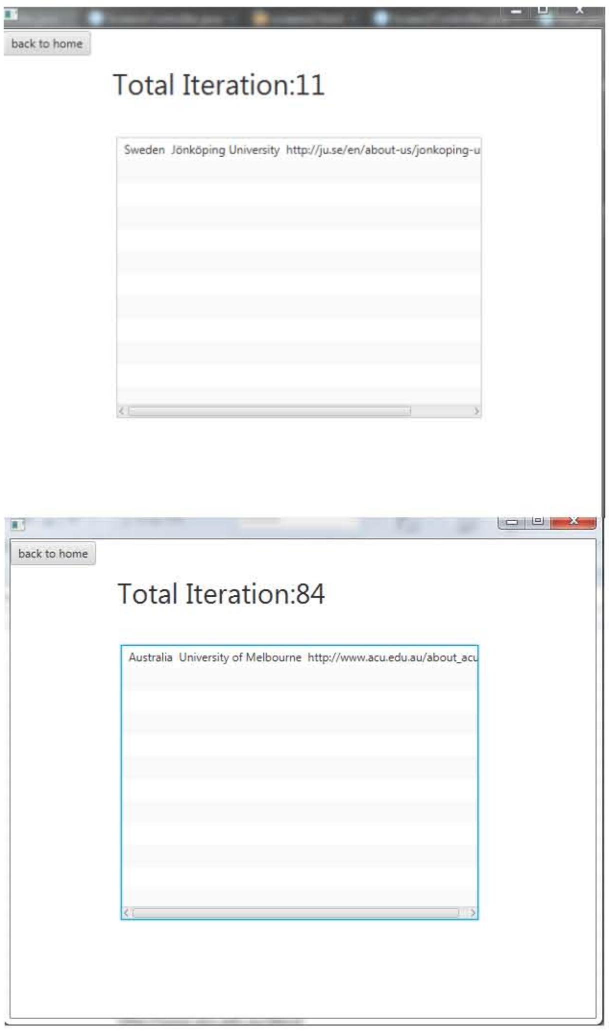
```
Bangladesh  Khulna University  http://ku.ac.bd/about-ku/
Bangladesh  Bangladesh Agricultural University  http://bau.edu.bd/pa
Bangladesh  Bangladesh University of Engineering and Technology  h
Bangladesh  Shahjalal University of Science & Technology  http://ww
Bangladesh  Univeristy of Dhaka  http://www.du.ac.bd/main_menu/th
Bangladesh  Bangladesh  Cse105
```

Total Iteration:144

```
Australia  Griffith University  https://bond.edu.au/about-bond
Australia  Edith Cowan University  http://www.ecu.edu.au/about-ec
Australia  Murdoch University   http://www.ecu.edu.au/about-ecu/
Australia  jiii  cse405
Australia  sydney  cse301.com
Australia  Macquarie University  http://www.csu.edu.au/about
Australia  Flinders University  http://about.curtin.edu.au/who/
Australia  jjjj  hdfkjhkdshkhadf
Australia  Swinburne University of Technology  http://www.deakin.
Australia  University Of Wollongong  http://about.curtin.edu.au/wh
Australia  Charles Sturt University  http://www.csu.edu.au/about
Australia  hhhhh  www.ispahani.about
```

Fig 4.4.1: search by location, before pre-processing and our optimized PSO.

If we select search by university and search with Jönköping University then it comes with the output in 11 iteration. In 423 dataset, it works normal as general PSO, it takes 84 iteration for output.



Total Iteration:11

```
Sweden  Jönköping University  http://ju.se/en/about-us/jonkoping-u
```

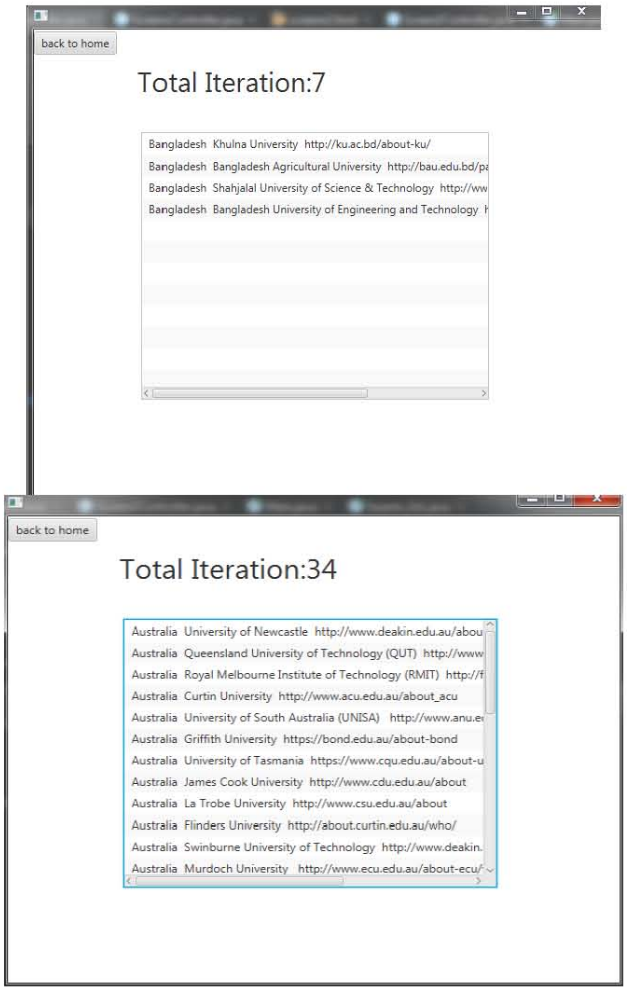
Total Iteration:84

```
Australia  University of Melbourne  http://www.acu.edu.au/about_acu
```

Fig 4.4.2: search by university, before pre-processing and our optimized PSO.

4.5 After Pre-processing and optimize PSO

Our Optimize PSO works very well when we apply it in after pre-processing data. It works perfectly when we search with location. We know location is always in low number but a location holds lot of data. In this algorithm When initialize the particles that time if we got the user request location then we don't go to the coding rest part ,we move to the .xlsl file because our data is in ordered by location . The Probability of getting location during initialization is very high. Now if we search by location Bangladesh the output comes in 7 iteration with 4 fresh Bangladesh data which is totally optimized. In 423 dataset it works perfectly in after preprocessing .Our optimize PSO actually based on the filtering/ordered data. Without ordering it works normal as general PSO. In 423 data set, it takes only 34 iteration to gather 30 fresh Australian University information, Here are the output:



Total Iteration:7

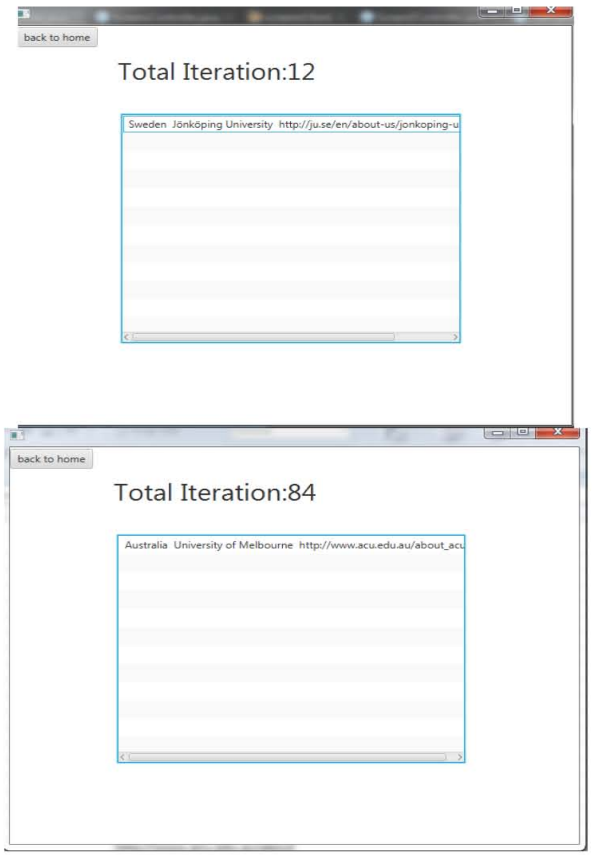
```
Bangladesh  Khulna University  http://ku.ac.bd/about-ku/
Bangladesh  Bangladesh Agricultural University  http://bau.edu.bd/pa
Bangladesh  Shahjalal University of Science & Technology  http://ww
Bangladesh  Bangladesh University of Engineering and Technology  h
```

Total Iteration:34

```
Australia  University of Newcastle  http://www.deakin.edu.au/abou
Australia  Queensland University of Technology (QUT)  http://www
Australia  Royal Melbourne Institute of Technology (RMIT)  http://f
Australia  Curtin University  http://www.acu.edu.au/about_acu
Australia  University of South Australia (UNISA)   http://www.anu.e
Australia  Griffith University  https://bond.edu.au/about-bond
Australia  University of Tasmania  https://www.cqu.edu.au/about-u
Australia  James Cook University  http://www.cdu.edu.au/about
Australia  La Trobe University  http://www.csu.edu.au/about
Australia  Flinders University  http://about.curtin.edu.au/who/
Australia  Swinburne University of Technology  http://www.deakin.
Australia  Murdoch University   http://www.ecu.edu.au/about-ecu/
```

Fig 4.5.1: Search by location, after pre-processing and our optimized PSO.

When we search by university on filtered data it not work as well as search by location but sometimes it also work as search by location. If we search by Jönköping University then it comes with the output in 12 iteration. In 423 dataset our Optimize PSO works as normal PSO. When user search by Australian Melbourne University it took 84 iteration. Sometime this Iteration number could be lower than 12. Here are the output screenshot.



Total Iteration:12

Sweden Jönköping University http://ju.se/en/about-us/jonkoping-u

Total Iteration:84

Australia University of Melbourne http://www.acu.edu.au/about_acu

Fig 4.5.2: Search by university, after pre-processing and our optimized PSO.

Chapter 5

Conclusion

5.1 Summary

The first practical application of PSO was in the field of neural network training and was reported together with the algorithm itself (Kennedy and Eberhart 1995). Many more areas of application have been explored ever since, including telecommunications, control, data mining, design, combinatorial optimization, power systems, signal processing, and many others. In our research, we have shown the differences between the general PSO & our modified one. Searching are more optimized than the general PSO. Not only that by using the pre-processing technique we could easily remove the irrelevant data from our dataset and we could easily remove duplicity.

5.2 Future work

Our future plan is to reuse the particle swarm optimization technique for reducing searching more. Not only that we are also interested to work the other fields of Particle Swarm Optimization for inventing something useful in future.

Appendix

In this chapter, we are providing the most important portions of the source code of our project in details.

```
private  voidPSOAlgorithm(introwCount) {
        Particle gBest = new Particle();
        Particle gBestTest;
        Particle aParticle;
int cycle = 0;
boolean done = false;
int x=0;
        List<Particle>resultPerticlelist=new
ArrayList<>();
stringArrayList.clear();
urlList.clear();
if(SearchButtonNoti!=null){
done=initializeParticle(rowCount / 10);
        }
else{
initializeParticle2(rowCount/10);
        }
while (!done) {
if (cycle < MAX_CYCLES) {
System.out.println("Size of particle
list"+particleList.size());
for (int i = 0; i <particleList.size(); i++) {
aParticle = particleList.get(i);
System.out.println(aParticle.getpBest());
if(aParticle.getpBest() == 3){
                          i=particleList.size();
                       }
gBestTest = findBest();
if (gBest.getpBest() <gBestTest.getpBest()){
gBest = gBestTest;
System.out.println(gBest.toString());
if(!stringArrayList.contains(gBest.toString())){
stringArrayList.add(gBest.toString());
urlList.add(gBest.geturl());
                        }
x++;
resultPerticlelist.add(gBest);
                    }
if(stringArrayList.size()>=10){
done=true;
                    }
if(chkSearchByuniversity!=null&&stringArrayList.size
()>=1){
done=true;
                    }
totalSearchCost++;
                }
updateParticle();
cycle++;
            }
else{
done = true;
            }
        }
    }
private  booleaninitializeParticle(intnParticle) {
particleList = new ArrayList<>();
boolean done;
done=false;
intrandom_index;
for (int i = 0; i <nParticle; i++) {
                Particle particle = new Particle();
int best = 0;
random_index = getRandomNumber();
particle.setData(getData(random_index));
particle.setIndex(random_index);
totalSearchCost++;
if (chkSearchBylocation != null) {
if (particle.getData()[0].equals(input)) {
best = 3;
System.out.println("searchbylocation not null");
match_index = particle.getIndex();
System.out.println("match at:"+match_index);
stringArrayList.add(particle.toString());
urlList.add(particle.geturl());
System.out.println("Add data:"+particle.toString());
```

```
AddmoreSameLocData();
done = true;
                        i=nParticle;
                    }
                } else if (chkSearchByuniversity !=
null) {
System.out.println("check by university not null
:D");
if (particle.getData()[1].equals(input)) {
best = 3;
System.out.println("searchbyuniversity not null");
match_index = particle.getIndex();
System.out.println("match at:"+match_index);
stringArrayList.add(particle.toString());
urlList.add(particle.geturl());
System.out.println("Add data:"+particle.toString());
done=true;
                        i=nParticle;
                    }
                }
particle.setpBest(best);
particleList.add(particle);
            }
System.out.println("Done from initialize
function:"+String.valueOf(done));
return done;
    }
private void AddmoreSameLocData(){
int index;
index=match_index+1;
for(int x=1;x<=10;x++){
            Particle particle=new Particle();
totalSearchCost++;
particle.setData(WhenMatchgetData(index));
particle.setIndex(index);
if(particle.getData()[0].equals(input)){
index=index+1;
stringArrayList.add(particle.toString());
urlList.add(particle.geturl());
}else {
index=match_index-(TotalNumof_thisCountry-x);
            }
```

```
        }
    }
private  intgetRandomNumber()
    {
return  (int)((rowCount - 1) * new
Random().nextDouble() + 1);


    }
private String[] getData(introw_index) {
String[] data = new String[3];
XSSFRow row;
XSSFCell cell;
row = sheet.getRow(row_index);
        Iterator cells = row.cellIterator();
int i = 0;
while (cells.hasNext()) {
cell=(XSSFCell) cells.next();
data[i] = cell.getStringCellValue();
i++;
        }
return data;
    }
private String[] WhenMatchgetData(int index) {
String[] data = new String[3];
XSSFRow row;
XSSFCell cell;
row = sheet.getRow(index);
        Iterator cells;
cells= row.cellIterator();
int i = 0;
while (cells.hasNext()) {
cell=(XSSFCell) cells.next();
data[i] = cell.getStringCellValue();
i++;
        }
return data;
    }
privateParticle findBest() {
        Particle result = particleList.get(0);
for (Particle aParticle : particleList) {
if (result.getpBest() <aParticle.getpBest())
```

```
result = aParticle;
        }
return result;
    }
privatevoid updateParticle(){
for (Particle aParticle : particleList) {
int best = 0;
intrandom_index;
random_index=getRandomNumber();
aParticle.setData(getData(random_index));
aParticle.setIndex(random_index);
if(chkSearchBylocation!=null){
if(aParticle.getData()[0].equals(input)){
best=3;


                }
            }
if(chkSearchByuniversity!=null){
if(aParticle.getData()[1].equals(input)){
best=3;
                }
            }
aParticle.setpBest(best);
        }
    }
public void preProcesingButtonHandle(ActionEvent
event) throws IOException {
freshParticleList = new ArrayList<>();
        List<String>stringList = new
ArrayList<String>();
        List<String>url = new ArrayList<String>();
inttotalRow,numOfDot,numOfSlash;
boolean done;
        String link;
done=false;
totalRow=readXLSXFileForFilter();
        List<String>uniqString = new
ArrayList<String>();
for (int i = 0; i <totalRow; i++) {
numOfDot=0;
numOfSlash=0;
link=null;
```

```
            Particle particle = new Particle();
int best = 0;
particle.setData(getData2(i));
particle.setIndex(i);
link=particle.getData()[2];
for(int z=0;z<link.length();z++){
if(link.charAt(z)=='.'){
numOfDot++;
                }
if(link.charAt(z)=='/'){

numOfSlash++;
                }
            }
if ((particle.getData()[2].contains("http://www.")||
particle.getData()[2].contains("http://")||
particle.getData()[2].contains("https://www.")||
particle.getData()[2].contains("https://"))&&numOfDo
t>=2&&numOfSlash>=3){
freshParticleList.add(particle);
            }
        }
uniqString.add(freshParticleList.get(0).getData()[0]
);
int position;
position=0;
for (int j=0;j<freshParticleList.size();j++){//uniq
string determine
if(!uniqString.contains(freshParticleList.get(j).get
Data()[0])){
uniqString.add(freshParticleList.get(j).getData()[0]
);


            }
        }
for (int j=0;j<uniqString.size();j++){//ordering
for(int x=0;x<freshParticleList.size();x++){
if(uniqString.get(j)==freshParticleList.get(x).getDa
ta()[0]){
stringList.add(freshParticleList.get(x).toString());
url.add(freshParticleList.get(x).geturl());
                }
            }
        }
System.out.println(uniqString);
ObservableList<String>urllistt;
ObservableList<String> list;
list =
FXCollections.observableArrayList(stringList);
urllistt=FXCollections.observableArrayList(url);
myController.setScreen(Main.screen2ID, "After
Filter", list,urllistt);
    }
public  intreadXLSXFile() throws IOException
    {
InputStreamExcelFileToRead = new
FileInputStream("pso_data.xlsx");
XSSFWorkbookwb = new XSSFWorkbook(ExcelFileToRead);
sheet = wb.getSheetAt(0);
returnsheet.getPhysicalNumberOfRows();
    }
    @FXML
public void psobasicBtnHandle(ActionEvent event)
throws IOException {
SearchButtonNoti=null;
mainwork();
ObservableList<String>urllistt;
ObservableList<String> list;
list =
FXCollections.observableArrayList(stringArrayList);
urllistt=FXCollections.observableArrayList(urlList);
chkSearchByuniversity=null;
chkSearchBylocation=null;
System.out.println("Total Search
cost"+totalSearchCost);
myController.setScreen(Main.screen2ID,
Integer.toString(totalSearchCost), list,urllistt);
}
```

References

[1] Zhan, Z-H.; Zhang, J.; Li, Y; Chung, H.S-H. (2009). "Adaptive Particle Swarm Optimization" (PDF). IEEE Transactions on Systems, Man, and Cybernetics. 39 (6): 1362–1381. doi:10.1109/TSMCB.2009.2015956.

[2] Yang, X.S. (2008). Nature-Inspired Metaheuristic Algorithms. Luniver Press. ISBN 978-1-905986-10-1.

[3] Tu, Z.; Lu, Y. (2004). "A robust stochastic genetic algorithm (StGA) for global numerical optimization". IEEE Transactions on Evolutionary Computation. 8 (5): 456–470. doi:10.1109/TEVC.2004.831258.

[4] Tu, Z.; Lu, Y. (2008). "Corrections to "A Robust Stochastic Genetic Algorithm (StGA) for Global Numerical Optimization". IEEE Transactions on Evolutionary Computation. 12 (6): 781–781. doi:10.1109/TEVC.2008.926734.

[5] Kennedy, James (2003). "Bare Bones Particle Swarms". Proceedings of the 2003 IEEE Swarm Intelligence Symposium.

[6] X. S. Yang, S. Deb and S. Fong, Accelerated particle swarm optimization and support vector machine for business optimization and applications, NDT 2011, Springer CCIS 136, pp. 53-66 (2011).
http://www.mathworks.com/matlabcentral/fileexchange/?term=APSO

[7] Parsopoulos, K.; Vrahatis, M. (2002). "Particle swarm optimization method in multiobjective problems". Proceedings of the ACM Symposium on Applied Computing (SAC). pp. 603–607.

[8] CoelloCoello, C.; Salazar Lechuga, M. (2002). "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization". Congress on Evolutionary Computation (CEC'2002). pp. 1051–1056.

[9] Roy, R., Dehuri, S., & Cho, S. B. (2012). A Novel Particle Swarm Optimization Algorithm for Multi-Objective Combinatorial Optimization Problem. 'International Journal of Applied Metaheuristic Computing (IJAMC)', 2(4), 41-57

[10] Kennedy, J. &Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm, Conference on Systems, Man, and Cybernetics, Piscataway, NJ: IEEE Service Center, pp. 4104-4109.