MAXIMIZING ADVERTISING VIEWS WITHIN A SPECIFIC BUDGET USING PARTICLE SWARM OPTIMIZATION

BY

ADITYA KAMAT NUID: 001383483

AKASH SRIVASTAVA NUID: 00138887

INDEX

What is Particle Swarm Optimization?

Particle Swarm Optimization is a problem solving technique which has been derived with respect to the behavior of flock of birds or fish.

Just as the birds fly together in a flock, every bird looks for the best position of the team, the ultimate goal and its best position and flies accordingly. Similarly, in particle swarm optimization, there are particles who move in a random direction initially and then with the help of a fitness function, the velocity of the particle gets updated with respect to its personal best, the team's best position and the global best position. In this way, all the particles work together in coordination with each other and try to find the optimal solution.

The velocity of the particle gets updated as per the formula given below

v[] = v[]*iF + cW * rand() * (pbest[] - present[]) + gW * rand() * (gbest[] - present[]) (a)

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

v[] is the particle velocity, persent[] is the current particle (solution). pbest[] and gbest[] are defined as the personal best and the global best. rand () is a random number between (0,1). cW, gW, iF are learning factors. cW is the Cognitive Weight, gW is the Global Weight and iF is the Inertia Factor

The above formula works, when there are a specific no of particles and they try to find the solution in accordance with each other. There is also a method of implementing particle swarm optimization which is Multi Swarm optimization. Here 1 swarm consists of particular number of particles and all the swarms work together in finding the optimal solution.

The velocity of particle in Multi Swarm Optimization is defined below

v[] = v[]*iF + cW * rand() * (pbest[] - present[]) + sW * rand() * (sbest[] - present[])+gW * rand() * (gbest[] - present[]) (a)

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

In this formula there is an additional factor called as the social best which is based on the best position of the swarm.

The pseudo code of particle swarm optimization is given below

```
For each particle
   Initialize particle
END

Do
   For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pBest) in history
         set current value as the new pBest
   End

   Choose the particle with the best fitness value of all the particles as the gBest
   For each particle
      Calculate particle velocity according equation (a)
      Update particle position according equation (b)
   End
```

The above pseudo code is for particle swarm optimization. If multi swarm is considered then the pseudo code will be similar except the velocity formula where the swarm best position is also considered along with the particle's personal best and global best position.

Particle Swarm Optimization is used in optimization problems, where we have to maximize the functions or it is also used in searching problems. In this project, particle swarm optimization finds its application in optimization problem.

Note: Out of n no of possible solutions, particle swarm optimization gives one of the optimal solutions. It is the optimal solution but that solution may not be the best one.

Significance of weights in PSO

The factors which are used in the formula of vector calculation are also referred to as weights. These weights are the Inertia Weight, Social Weight, Cognitive Weight and the Global Weight. Depending upon the problem which is being solved by PSO and depending upon the no of particles and swarm, the values of these weights could be changed in order to give the best optimal solution in lesser amount of time.

There are two concepts in PSO, known as the exploration and the exploitation. Exploration means the particles try to cover maximum area where they want to search the target and exploitation means particle try to expose a particular area in depth. Depending upon the values of the weights, the way to solve the problem can either be carried by the means of exploration or by exploitation.

When the inertia weight is increased then the performance of PSO is more towards exploratory path.

Cognitive weight represents the individual weight of the position that was best for the particle. The effect of this weight is that the particle is again drawn towards its best position.

Social weight quantifies the performance of the particle with respect to the group of particles. The global weight quantifies the performance of the particle with respect to the best position among all.

In this project, linear decreasing strategy has been adopted where it gives efficient results. Experimentally, it has been proven that the PSO performs best when the inertia weight is between 0.4 and 0.9. After trying out various values of inertia weight, 0.79 worked best in this project.

The values of cognitive weight, social weight and global weight have been decided experimentally and with the references and help of 2 research papers based on implementation of weights in PSO.

Inertia weight: 0.79

Cognitive weight: 0.3

Social Weight: 0.01

Global Weight: 0.9

Usage of PSO in Optimization problems

In optimization process, there are a many solutions available but optimized solution or the best solution is required with respect to a number of factors. These factors could be finding the solution with minimum cost which is usually used in economic dispatch problem. Other applications are in the areas of electronics, civil engineering, health analysis and genetics.

In Optimization the particles try to converge towards the solution which satisfies the criteria of an optimized solution. These formulas or equation are mentioned in the fitness function.

Hence fitness function play a very important role in deciding the convergence point and hence, the best solution.

Implementation of PSO in maximizing the views in specified budget

The problem, which has been selected in this project, is an optimization problem only. The problem is described as follows

Consider there are 3 sources of advertising and every source gives its quotation for displaying advertisement on its medium along with the expected no views it will get for its advertisement.

The marketing company has its fixed budget. In that budget, the company needs to find the best optimal solution in terms of number of advertisements it will post on each website in order to get the maximum no of views.

There will be two equations which will be used in the fitness function

Budget = (No of ads in Web A)*(Cost of A)+(No of ads in Newspaper)*(Cost of B)+No of ads in Movie Theatre)*(Cost of C)

Max Views = (No of ads in Web A)*(Views in A)+ (No of ads in Newspapers)*(Views in B)+ (No of ads in Movie Theatre)*(Views in C)

The swarms will find the values of no of ads in sources A, B and C such that it will give the maximum number of views of ads in specified budget.

The fitness function devised is dynamic in nature that means it can be used for calculating the max no of views starting from 2 to 6 sources. The no of sources/mediums can be increased by changing the fitness function but due to limited computational power, it has been kept 6.

Marketing budget allocation has always been of major concern as companies' expenditure on marketing is large and hence the companies always expect a lot of returns from marketing. Hence it is necessary that the budget it spends on marketing is utilized fully and advertisement reaches to maximum no of people

Use of Multi threading and TimerTask Class

The java Timertask class must be used to simulate the concurrent computations. TimerTask class defines a task that can be scheduled to run for just once or for repeated number of time. In order to define a TimerTask object, this class needs to be implemented and the run method needs to be overridden. The run method is implicitly invoked when a timer object schedules it to do so.

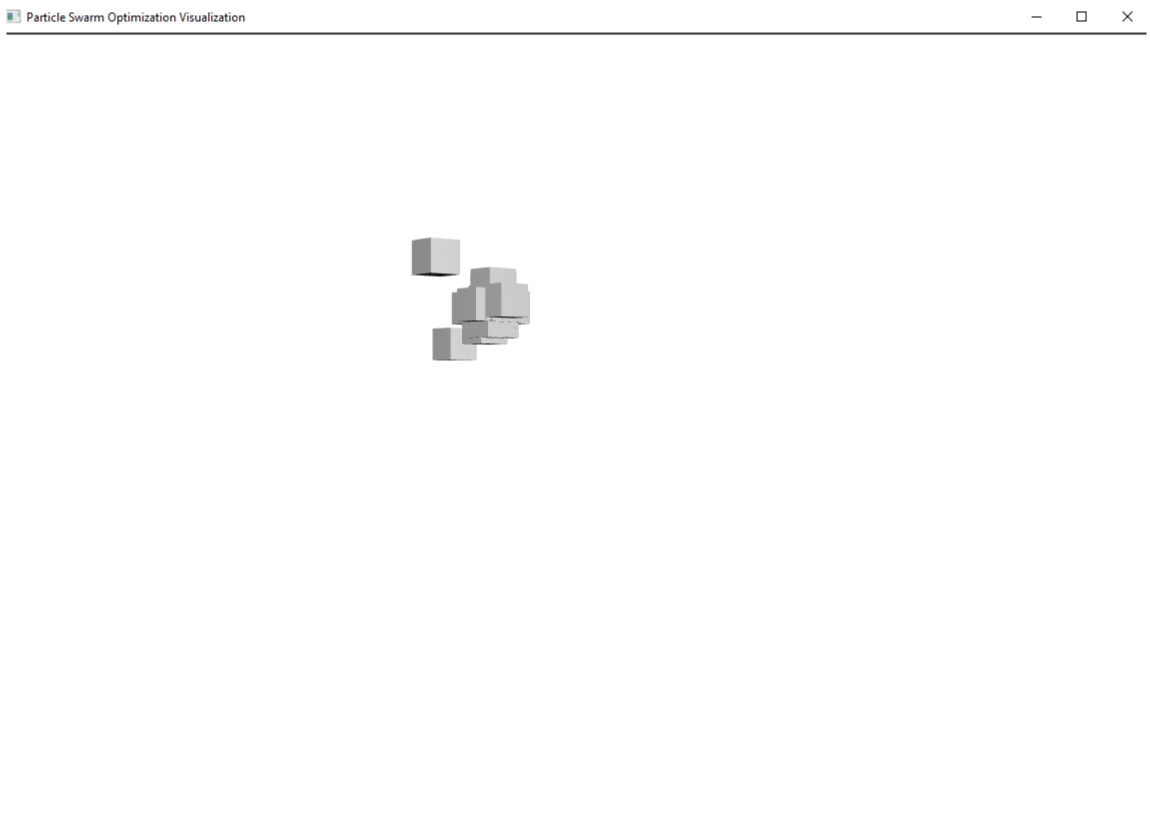
The TimerTask has been used in order to run the iterations of particle swarm optimization. Iteration is carried out on a separate thread using the timertask. It has been scheduled with a delay of 400-700 ms where each of iteration is carried out after the mentioned delay. This adds to concurrent computation and increases the efficiency of algorithm.

There is another thread which is used to run the animation. It uses a sleep method which adds a delay of 100 ms after iteration in order to avoid any exceptions.

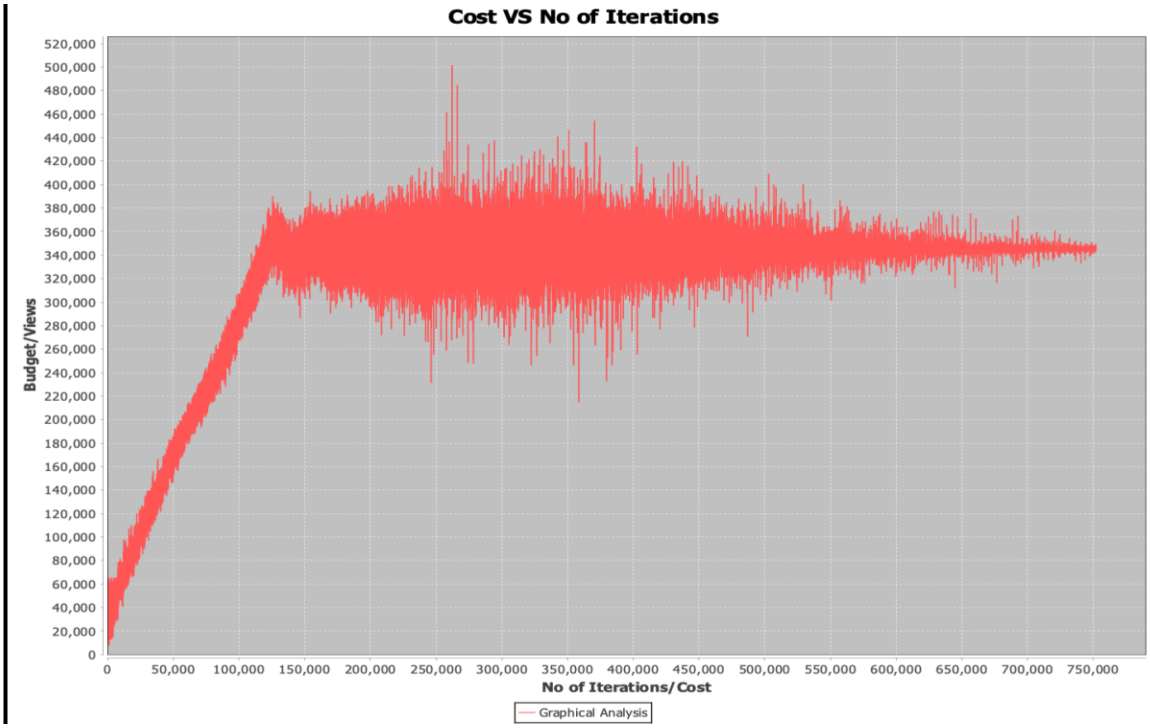
Usage of Graphical Analysis and Animation

In order to visualize the movements of particles, animation has been implemented using the Java FX application and its components. The 3D animation shows the movements of the particles in 3D domain

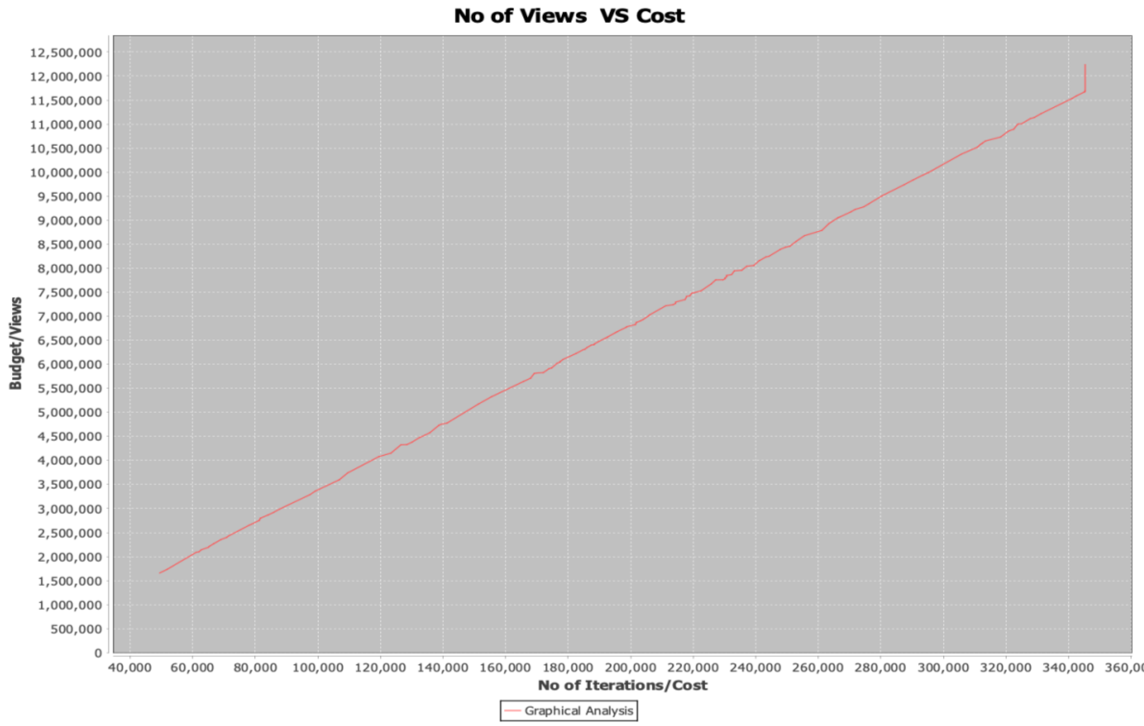
As it can be observed from the above image, every cube represents one particle. Initially all the particles are scattered randomly and slowly, all the particles start to converge at one point which is the best position for maximizing the total no of views.



In this image, it can be seen that the particles have converged to a great extent than the previous one.



There are two graphs which show the output in order to understand the PSO in a better way. The first graph shows the output of cost vs the no of iterations. This graph show that initially the particles are scattered randomly and hence every particle has random position because of this, the initial budget is extravagantly large. But, slowly as the particles tries to converge, it tries to achieve the required budget.



The second graph is plotted between the no of views vs the required budget. Here the budget is constant and the no of views changes. This states, how the particles tries to converge at a point where it maximizes the no of views for the said budget

Major Challenges faced

Particle Swarm Optimization largely depends on many factors like the inertia factor, the cognitive weight, social weight and global weight. It is necessary to decide a right set of values otherwise it may give a wrong output or it may take a large amount of time to converge and give the right output.

Particle swarm optimization also depends on the no of particles. If the no of particles are less then, if exploratory analysis is needed then it may not be possible for the particles in order to explore that much area.

Because of these issues, it may be required to change these factors and even the no of particles for some of the scenarios and re-run the algorithm.

Conclusion

Particle Swarm Optimization is simple to understand works efficiently for solving different types of optimization and search problems provided that the right factors are passed to it. If combined with TimerTask and Multi threading, it becomes more efficient because of concurrent computations

References

1- https://www.geeksforgeeks.org/java-util-timertask-class-java/

2- Inertia Weight Strategies in Particle Swarm Optimization https://ieeexplore.ieee.org/document/6089659

3- Fundamentals of Computational Swarm Intelligence by Engel Brecht https://web2.qatar.cmu.edu/~gdicaro/15382/additional/CompIntelligence-Engelbrecht-ch16.pdf

4- https://en.wikipedia.org/wiki/Particle_swarm_optimization

5- https://shodhganga.inflibnet.ac.in/bitstream/10603/28526/9/09_chapter4.pdf

Contribution:
50% by Aditya Kamat
50% by Akash Srivastava