**Project Report on Dynamic Pricing using Particle Swarm Optimization**

Gaurang Deshpande

001647112

deshpande.ga@husky.neu.edu

**Problem Statement**

The objective of this project is to understand, practice and implement the swamp computing approach to find out the best price for products based on various factors in e-commerce application to maximize the revenue. Dynamic pricing is a strategy to modify the price of commodity over time, depending on various market scenarios, to obtain the optimum revenue.

A firm can improve their revenues be dynamically adjusting price of product rather than adopting fixed price throughout product life. The factors that can affect the price of a product can be anything that shifts the demand characteristics like seasons, marketing, advertisement, rival advertisements, etc.

The idea is to evaluate the market factors affecting the prices as the objective function used in particle swamp optimization model to find out particle best and global best for entire swamp.

Because there exists a problem of finding or considering such hidden parameters, while exploring the best prices, PSO is an interesting approach.

**Use Cases**

Finding optimum revenue for product depending on holidays, different time of year, quantity demand, variable cost changes with quantity of product to be sold, weather, etc.

Modifying the predefined conditions and parameters and observing the results.

Deriving the conclusions about pricing dependencies

**Particle Swarm Optimization:**

One of the hottest areas in artificial intelligence is what is known as Swarm Computing –a unique approach to using local knowledge and decisions to solve global optimization problems.

PSO is an interesting option to explore to solve our problem at hand as it tends to naturally operate at the boundary between stability and chaos.

Particle swarm optimization (PSO) is a computational method that [optimizes](https://en.wikipedia.org/wiki/Mathematical_optimization) a problem by [iteratively](https://en.wikipedia.org/wiki/Iterative_method) trying to improve a [candidate solution](https://en.wikipedia.org/wiki/Candidate_solution) with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed [particles](https://en.wikipedia.org/wiki/Point_particle), and moving these particles around in the [search-space](https://en.wikipedia.org/wiki/Optimization_(mathematics)#Concepts_and_notation) according to simple [mathematical formulae](https://en.wikipedia.org/wiki/Formula) over the particle's [position](https://en.wikipedia.org/wiki/Position_(vector)) and [velocity](https://en.wikipedia.org/wiki/Velocity). Each particle's movement is influenced by its local best known position, but is also guided toward the best-known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is implemented per following equations:

Vi+1 = in \* Vi + pac \* r1 \* (Pb - Pi) + gac \* r2 \* (Pg - Pi)

And

Pi+1 = Pi + Vi+1

where:

Vi = Current Velocity

Vi+1 = Next Velocity

Pi = Current Position

Pi+1 = Next Position  
Pb = Personal Best of particles

Pg = Global Best of all particles  
in = Inertia coefficient  
pac = Personal Acceleration Coefficient for particles  
gac = Global Acceleration Coefficient for particle  
r1 & r2 = Random

In PSO, one of the important steps is to calculate cost function. According to cost function, the particles accumulate near global best.

Cost function for this model is

d = e al + p \* beta + noise

here

d = demand

p = price

al, beta and noise are factors dependent on market values like marketing, advertisement, rival advertisements, etc.

beta is strictly negative

Revenue for a product is calculated as r = p\*d

PSO is implemented as each particle as a random product with random value of price and for each iteration, revenue is calculated as per cost function. For each iteration, global best and personal best for each particle is calculated.

The Timertask class was implemented to make all particles compute their personal best and overall global best concurrently.

JFreeCharts are implemented to show the position for all particles in research space. The plot shows price vs revenue calculated at that price as per cost function described above. Through all iterations, the same plot is implemented with same bounds with different particle positions, creating an animation.

To identify the dependency of revenue on all the parameters described in cost function, we modified the parameters during computation and the effect on price and global best is observed.

**Findings and Conclusion:**

* PSO is implemented to calculate the prices for product which can give maximum revenue.
* After running the algorithm, it can be inferenced that price is independent of al and noise.

It is only dependent on beta.

* A graph / animation can be seen where we will plot price vs revenue and can be inferenced that all particles will move towards global best revenue.

**Reference:**

Particle Swarm Optimization in Dynamic Pricing

By Patrick B. Mullen, Christopher K. Monson, Kevin D. Seppi and Sean C. Warnick

Computer Science Department

Brigham Young University

