Savitribai Phule Pune University

A MINI PROJECT REPORT

ON

APPLY THE PARTICLE SWARM OPTIMIZATION FOR TRAVELLING SALESMAN PROBLEM

Submitted by

Swati Patra	**71710572F**
Tukaram Rathod	**71612446H**
Hrishikesh Dabir	**71710273E**

Under the guidance of

Dr. A.P. Kale

Department of Computer Engineering

Modern Education Society's

College Engineering, Pune-411 001

2019-20

C E R T I F I C A T E

This is to certify that the Project Entitled

Apply The Particle Swarm Optimization for Travelling salesman problem

Submitted by

Swati Patra	71710572F
Tukaram Rathod	71612446H
Hrishikesh Dabir	71710273E

is a bonafide work carried out by Students under the supervision of Dr. A.P. Kale and it is submitted towards the partial fulfillment of the requirement of Bachelor of Engineering (Computer Engineering).

(Dr. A.P. Kale)
Project Guide,
Department of Computer Engineering

Place : MESCOE, Pune

Date :

ABSTRACT

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and 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.The Particle Swarm Optimizer employs a form of artificial intelligence to solve problems. It is particularly good at finding solutions to functions that use multiple, continuously variable, values. This piece is concerned with modifying the algorithm to tackle problems, such as the travelling salesman problem, that use discrete, fixed values.

Keywords: Particle Swarm OPtimization, Travelling Salesman Problem, Brute Force, Search Space.

CHAPTER 1
INTRODUCTION

1.1 Particle Swarm Optimization

Particle swarm optimization, PSO, is an evolutionary computation technique inspired in the behavior of bird flocks. PSO algorithms were first introduced by Kennedy & Eberhart (1995) for optimizing continuous nonlinear functions. The fundamentals of this metaheuristic approach rely on research where the movements of social creatures were simulated by computers.PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found. Also, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods.

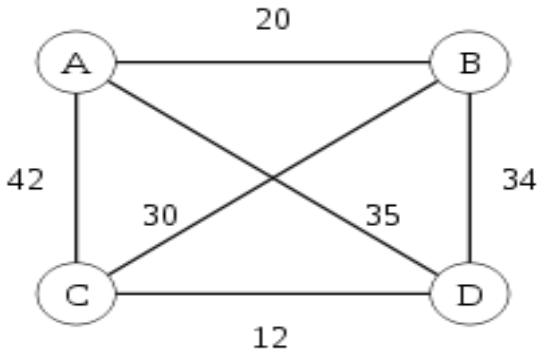
A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

1.2 Travelling Salesman Problem

The travelling salesman problem (also called the travelling salesperson problem[1] or TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization.

TSP can be modelled as an undirected weighted graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's weight. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once. Often, the model is a complete graph (*i.e.* each pair of vertices is connected by an edge). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour.

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept *city* represents, for example, customers, soldering points, or DNA fragments, and the concept *distance* represents travelling times or cost, or a similarity measure between DNA fragments.



CHAPTER 2
METHODOLOGY

The basic idea is to move (fly) a group (swarm) of problem solving entities (particles) throughout the range of possible solutions to a problem. This range is known as the problem space. The movement of particles within the problem space has a random component but is mainly guided by three factors.

1. The present position of the particle.
2. The best position found by the particle, known as personal best or pBest.
3. The best position found in the swarm, known as a global best or gBest.

Modern variations of the algorithm use a local best position rather than a global best. This tends to ensure better exploration of the problem space and prevents too rapid a convergence to some regional minimal value. In these variations, the swarm is divided into groups of particles known as informers. Information is exchanged between every member of a group to determine the local best position for that group The particles are reorganised into new groups if a certain number of iterations pass without the global best value changing.

The Original PSO Formula

The formula for dealing with continuously variable, values is

Vid=vid*W+C1*rand(pid-xid)+C2*Rand(pgd-xid)

where

vid is the current velocity and **Vid** is the new velocity. The velocity, in this case, is the amount by which the position is changed.

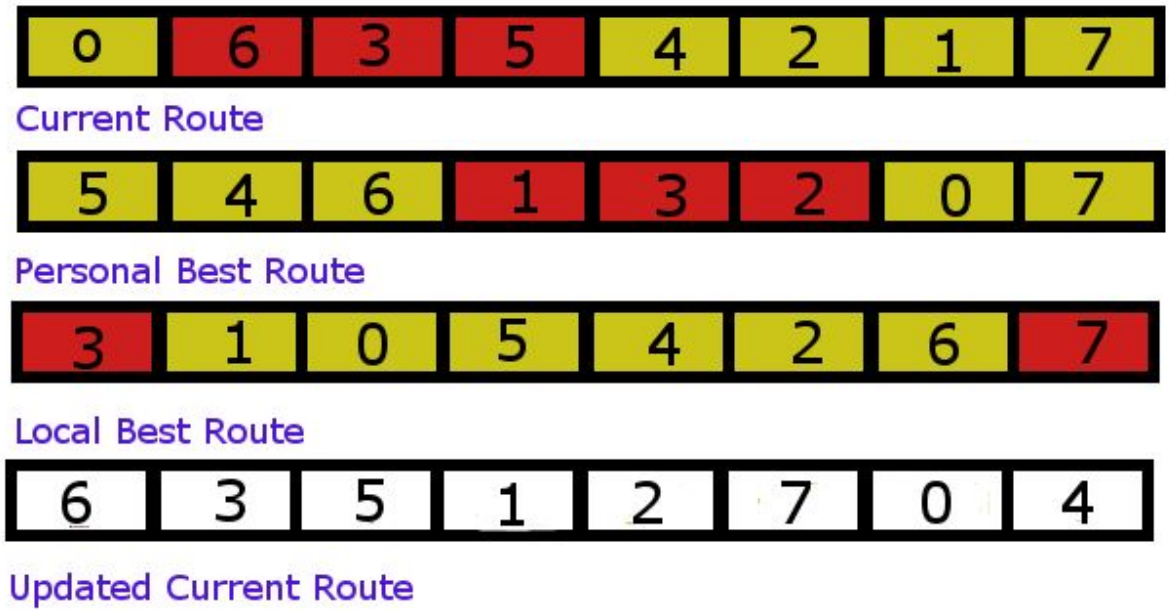
W, C1,C2 are constants. The approximate values for the constants are C1=C2=1.4 W=0.7

Rand and **rand** are two randomly generated doubles >=0 and <1

xid is the current position, **pid** is the personal best position and **pgd** is the global best position. The position is then updated by adding the new velocity to it. **xid=xid+Vid**. This formula is applied to each dimension of the position.

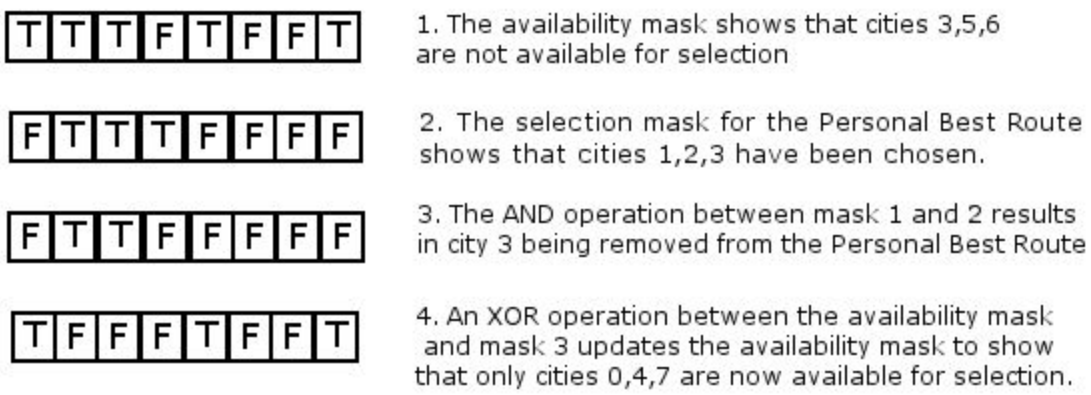
Using a PSO to Update the Salesman's Route

As we have seen, the new position of a particle is influenced to varying degrees by three factors. They are the particle's present position, its best previous position and the best position found within its group. The salesman's route can be updated by dividing it into three sections, one for each of the three factors, where the size of each section is determined by that section's relative strength. The sections can then be joined together to form an updated route. But there is a problem with this approach. Cities can only be listed once and sections may contain cities that have already been listed in a previous route section. So there needs to be a mechanism to ensure that every city is added to the route and that no city is duplicated in the process.

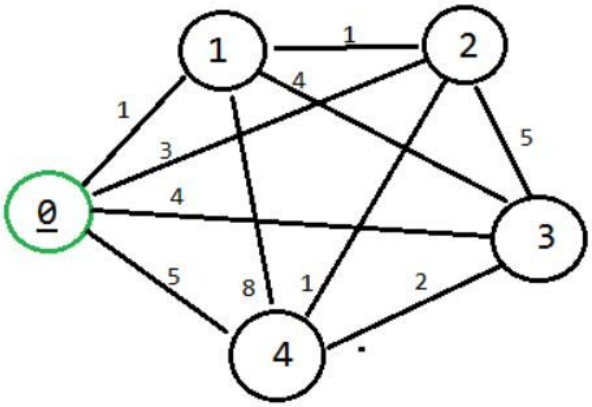


In the diagram above, the section selected from the Current Route is 6,3,5. These cities are added to the new route. The Personal Best Route has the section 1,3,2 selected. Selection 3 has already been added, so only cities 1 and 2 are added. The Local Best Route has section 7,3 selected. City 3 has already been added so only city 7 gets selected. Finally, the two cities that have not been selected, cities 0 and 4, are added to the new route in the order that they appear in the Current Route.

The selection of cities to be added is facilitated by using BitArrays. One BitArray is used as an availability mask with all the bits being set initially to true. Another BitArray is used as a Selection Mask for the segment to be added. To illustrate this, consider the situation after the Current Segment has been added.



TSP to be solved



RESULT

```
Showing particles...

pbest: [0, 3, 4, 2, 1]  |   cost pbest: 9   |   current solution: [0, 4, 2, 3, 1]   |   cost current solution: 16
pbest: [0, 2, 4, 3, 1]  |   cost pbest: 11  |   current solution: [0, 2, 4, 3, 1]   |   cost current solution: 11
pbest: [0, 1, 2, 4, 3]  |   cost pbest: 9   |   current solution: [0, 3, 2, 1, 4]   |   cost current solution: 23
pbest: [0, 2, 4, 3, 1]  |   cost pbest: 11  |   current solution: [0, 2, 4, 3, 1]   |   cost current solution: 11
pbest: [0, 4, 1, 3, 2]  |   cost pbest: 25  |   current solution: [0, 4, 1, 3, 2]   |   cost current solution: 25
pbest: [0, 3, 4, 2, 1]  |   cost pbest: 9   |   current solution: [0, 3, 4, 2, 1]   |   cost current solution: 9
pbest: [0, 4, 3, 2, 1]  |   cost pbest: 14  |   current solution: [0, 4, 3, 2, 1]   |   cost current solution: 14

gbest: [0, 3, 4, 2, 1] | cost: 9
```

CHAPTER 5
CONCLUSION

Particle swarm optimizer approach works very well on the Traveling Salesman Problem. That approach significantly reduced the number of visited routes to find the optimal route, compared with the Brute-Force Search. Although our sample problem is not so big, it reduced the execution time in a significant amount. When the problem gets bigger, the execution time benefits can be much more significant. The performance measures can differ a lot with simple optimizations on implementation decisions. Giving directions to a searching quest of an optimal path should be almost every time better than just searching blindly. To summarize, A Particle swarm optimizer can be used to solve highly complicated problems by multiple repetitions of a simple algorithm.given the results of this experiment, Particle swarm optimizer seem to be good solutions on NP-Complete problems.