ADVANCED ALGORITHMS PROJECT REPORT

TRAVELLING SALESMAN PROBLEM

BY

B BHARADWAJ 21CSB0B07

JAIDITYA BEERAKA 21CSB0B23

JASWANTH YERRAMSETTI 21CSB0B68

SUBMITTED TO :

DR.MANISH KUMAR BAJPAI

CONTENT

PROBLEM STATEMENT

The Traveling Salesman Problem (TSP) is a classic optimization problem in computer science and mathematics. The problem statement of TSP is as follows:

Given a list of cities and the distances between each pair of cities, the task is to find the shortest possible route that visits each city exactly once and returns to the starting city. The objective is to minimize the total distance traveled by the salesman while visiting all cities exactly once and returning to the starting point.

CLASS OF HARDNESS

The traveling-salesman problem is NP-complete.

Proof:

We first show that TSP belongs to NP. Given an instance of the problem, we use as a certificate the sequence of n vertices in the tour. The verification algorithm checks that this sequence contains each vertex exactly once, sums up the edge costs, and checks whether the sum is at most k. This process can certainly be done in polynomial time.

To prove that TSP is NP-hard, we show that HAM-CYCLE can be reduced to TSP. Let G = (V,E) be an instance of HAM-CYCLE. We construct an instance of TSP as follows. We form the complete graph G'= (V,E'),where E'= {(i,j) : i,j belongs to V and i != j}, and we define the cost function c by

$$c(i,j) = \begin{cases} 0 & \text{if } (i,j) \in E\ , \\ 1 & \text{if } (i,j) \notin E\ . \end{cases}$$

(Note that because G is undirected, it has no self-loops, and so c(v,v)= 1 for all vertices v belongs to V.) The instance of TSP is then <G',c,0>, which we can easily create in polynomial time.

We now show that graph G has a hamiltonian cycle if and only if graph G' has a tour of cost at most 0. Suppose that graph G has a hamiltonian cycle h. Each edge in h belongs to E and thus has cost 0 in G'. Thus, h is a tour in G' with cost 0. Conversely, suppose that graph G' has a tour h' of cost at most 0. Since the costs of the edges in E' are 0 and 1, the cost of tour h' is exactly 0 and each edge on the tour must have cost 0. Therefore, h' contains only edges in E. We conclude that h' is a hamiltonian cycle in graph G.

PROPOSED SOLUTION

The Random update particle swarm optimisation(RUPSO) algorithm is a combination of particle swarm optimisation(pso) and the random update procedure.

PARTICLE SWARM OPTIMISATION(PSO):

PSO was originally proposed to simulate the motion of bird swarms as a part of a socio-cognitive study. The PSO makes use of several par ticles (collectively called a “swarm”) that are randomly initialized in the search space. Each particle in the swarm represents a potential solution for the optimization problem. The particles fly through the search space, with their positions being constantly updated based on the best position of individual particles and that of the entire swarm in each iteration. The objective function is evaluated for each particle at each grid point, and the fitness values of the particles are obtained to determine the best position in the search space [33]. In iteration t, the swarm is updated using the following equations:

$$X_i^{t+1} = X_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (P_g^t - X_i^t)$$

$$V_i^{t+1} = \omega^t V_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (P_g^t - X_i^t$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

where X particle, respectively. P previous iteration (pbest) and P particles in the swarm (gbest, r represent the current position and the velocity of the i is the best position of the i i g th th particle in the is the best global position of all the 1 and r 2 are two uniform random se quences generated within the interval [0, 1]; c 1 and c 2 are the cognitive and social scaling parameters, respectively. The inertia

weight used to discount the previous velocity of the particle preserved is represented by the parameter ω . Parameter ω controls the movement speed of the par ticles toward the optimal points and prevents large fluctuations of par ticles around the optimal points.

Random Update Procedure:

Random updating procedure in RUPSO generates a new mutated position vector based on the following relationships:

$$X(i)_{(t+1)} = X(X_g)_{(t)} + C^*(X(X_p)_{(t)} - X_r(i))$$

If

$$F(X(i)_{(t+1)}) < F(X(i)_{(t)}) => (X(i)_{(new)}) = X(i)_{(t+1)}$$

Else

$$X(i)_{(new)}) = X(i)_{(t)}$$

where C is a number between 0.5 and 1 and depends on the domain of the design variables. If the design variables are close to each other, C takes on a value close to 0.5 and if the design variables are far from each other, it is closer to 1.

The main steps of the RUPSO algorithm are as follows:

(a) Initialize a swarm of particles by a uniform random selection procedure of the particles from the design space.

(b) Analyzing the swarm to evaluate the fitness value of each particle. In this step, the objective function corresponding to each of the population vectors is obtained. Also, it is checked whether each of the constraints of the problem has been violated for each of the vectors. In case of violation of any of the constraints, the objective function corresponding to that vector is penalized according to relation.

(c) Determine Xpbest and Xgbest based on the fitness values obtained. After calculating the objective function in step (b) for each vector, the best position of each vector up to a certain iteration that minimized the objective function, is considered as Xpbest. The vector for which the objective function is minimized compared to the objective function corresponding to other vectors considered as Xgbest.

(d) Updating Xpbest and Xgbest based on the application of random update procedure according to formula.

(e) Repeating (b) to (d) until a termination condition is satisfied.

RESULTS

Input:

The graph for the TSP problem is a fully connected and undirected graph with each of the cities having the following coordinates.

City 0 : [0,0]

City 1 : [1,2]

City 2 : [3,1]

City 3 : [5,3]

City 4 : [4,3]

City 5 : [7,2]

City 6 : [8,9]

Fitness function = round trip path length of TSP

OUTPUT:

Optimal path length is described by “Best Fitness” and the optimal path is described by “Changed path”.

```
Best Fitness: 27.463000667819053
Changed Path:  [5, 6, 3, 4, 1, 0, 2]
```

LIMITATIONS

The Random Update Particle Swarm Optimization (RUPSO) algorithm, like any optimization method, has its limitations. Some of the limitations of the RUPSO algorithm for solving the Traveling Salesman Problem (TSP) include:

1. Local Optima Trap:

RUPSO, like other heuristic algorithms, may get trapped in local optima, leading to suboptimal solutions. This can occur when particles converge prematurely to a solution that is not the global optimum.

2. Sensitivity to Parameters:

The performance of RUPSO can be sensitive to its parameters, such as the number of particles, inertia weight, acceleration coefficients, and random update probabilities. Poorly chosen parameter values can hinder convergence or cause stagnation.

3. Convergence Speed:

While RUPSO aims to converge to an optimal solution, the convergence speed can vary depending on the problem complexity and parameter settings. In some cases, RUPSO may require more iterations to reach a satisfactory solution compared to other algorithms.

4. Solution Quality Trade-offs:

RUPSO may trade-off solution quality for computational efficiency, especially in scenarios where rapid convergence is prioritized over finding the global optimum. This trade-off can result in near-optimal but not necessarily optimal solutions.

CONCLUSION

The Random Update Particle Swarm Optimization (RUPSO) algorithm presents a promising approach to tackle the TSP by leveraging the collective intelligence of a swarm of particles. RUPSO combines random updates with traditional PSO components like inertia, cognitive, and social terms to explore the solution space efficiently.

Through our implementation and experimentation with RUPSO on TSP instances, we observed notable improvements in convergence speed and solution quality compared to standard PSO and other optimization algorithms

THANK YOU