**21CS2204RA – MATHEMATICAL PROGRAMMING**

**A Project Report**

**on**

**TRANSPORTATION NORTH WEST METHOD**

**PARTICLE SWARN OPTIMIZATION**

**Under the Guidance of**

# **Mr.Sanjay Yadav**

# Professor, Department of CSE-H

**by**

I.D NUMBER NAME

**2100030130 DHANYA SRI GRANDHI**

**A close-up of a logo

Description automatically generated with low confidence**

**KONERU LAKSHMAIAH EDUCATION FOUNDATION**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(DST-FIST Sponsored Department)**

Green Fields, Vaddeswaram, Guntur District-522 502

April - 2023

**Declaration**

We hereby declare that this Project report entitled **TRANSPORTATION NORTH WEST METHOD, PARTICLE SWARN OPTIMIZATION** has been prepared by us in the course **21CS2204 MATHEMATICAL PROGRAMMING** in **COMPUTER SCIENCE AND ENGINEERING** during the Even Semester of the academic year 2022-2023. We also declare that this project-based lab report is our own effort.

**Date:04/05/2023**

**Place:**Vijayawada

Signature of the Student

Id Number Student Name

2100030130 DHANYA SRI GRANDHI

**CERTIFICATE**

This is to certify that the project-based Lab report entitled **“TRANSPORTATION NORTH WEST METHOD, PARTICLE SWARN OPTIMIZATION”** is a bonafide work done by Ms. **DHANYA SRI GRANDHI** bearing Regd. No. **2100030130** to the course **21CS2204 MATHEMATICAL PROGRAMMING** in COMPUTER SCIENCE AND ENGINEERING during the Even Semester of the Academic year 2022-2023.

**FACULTY IN CHARGE**

**Mr.Sanjay Yadav**

**ACKNOWLEDGEMENTS**

Our sincere thanks to **Mr.Sanjay Yadav** in the lab sessions for his/her outstanding support throughout the project for the successful completion of the work.

We express our gratitude to **Dr. Inunganbi Sanasam** Course Co-Ordinator for the course 21CS2204RA Mathematical programming in the Department of Computer Science and Engineering for providing us with adequate planning and support and means by which we can complete this project.

We express our gratitude to **Prof. A. Senthil,** Head of the Department for Computer Science and Engineering for providing us with adequate facilities, ways and means by which we can complete this project.

We would like to place on record a deep sense of gratitude to the Vice Chancellor of K L University for providing the necessary facilities to carry out the project.

Last but not least, we thank all Teaching and Non-Teaching Staff of our department and especially our classmates and our friends for their support.

Name: DHANYA SRI GRANDHI

Reg. No: 2100030130

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
| 1 | Abstract | 6 |
| 2 | Introduction | 7 |
| 3 | Literature Work | 8 |
| 4 | System Requirement Specification | 9 |
| 5 | Methodology | 10 |
| 6 | Coding and implementation | 13 |
| 7 | Result Analysis | 14 |
| 8 | Conclusion | 15 |
| 9 | Future Enhancement | 16 |
| 10 | Bibliography | 17 |

**ABSTRACT**

The transportation problem involves determining the optimal allocation of goods from suppliers to demand points. The North West Corner Method is used to find an initial feasible solution by starting from the top-left corner of the cost matrix and allocating as much as possible to the corresponding supply and demand cells. This method ensures that the allocations are made in a systematic manner while considering the supply and demand constraints.

Particle Swarm Optimization (PSO) is a metaheuristic algorithm inspired by the behavior of bird flocks or fish schools. It involves a population of particles that move through the search space to find the best solution. Each particle has a position and a velocity, which are updated based on its own best-known position and the best-known position of the entire swarm.

During the optimization process, particles communicate with each other by sharing information about their own best-known positions and the global best-known position discovered by any particle in the swarm. This information exchange allows the particles to collectively move towards promising regions of the search space.

**INTRODUCTION**

The transportation problem is a classic optimization problem in mathematical programming that deals with the allocation of goods from a set of suppliers to a set of demand points, while minimizing the total transportation cost. The North West Corner Method is a popular heuristic used to find an initial feasible solution for the transportation problem.

Particle Swarm Optimization (PSO) is a nature-inspired optimization algorithm that is based on the behavior of a swarm of particles. Each particle represents a potential solution in the search space, and the swarm collectively explores the solution space to find the optimal solution.

**LITERATURE WORK**

Hillier, F. S., & Lieberman, G. J. (2010). Introduction to operations research. Tata McGraw-Hill Education. This textbook provides a comprehensive introduction to operations research, including transportation problems and initial solution methods such as the North-West Corner Method. The book provides clear explanations of the methodology, advantages, and limitations of the North-West Corner Method, along with practical examples and exercises. Ragsdale, C. T. (2014).

Spreadsheet modeling and decision analysis: a practical introduction to business analytics. Cengage Learning. This book provides a practical introduction to spreadsheet modeling and decision analysis, including transportation problems and the North-West Corner Method. The book provides step-by-step instructions for implementing the North-West Corner Method in Excel, along with practical examples and exercises.

**SYSTEM REQUIREMENT SPECIFICATION**

* **SOFTWARE REQUIREMENTS:**

The major software requirements of the project are as follows:

Language : Python, Python Libraries

Tools **:** Microsoft Word and Jupiter Notebook

* **HARDWARE REQUIREMENTS:**

The hardware requirements that map towards the software are as follows:

* Intel  (or AMD equivalent) i5 or better processor, 7th generation or newer (Virtualization must be supported)
* Windows 10 Operating System
* 1920 x 1080 or greater screen resolution
* 500 GB or larger SSD
* Minimum 8 GB of RAM (12GB -16GB RAM recommended)
* Access to High Speed Internet

**METHODOLOGY**

Begin by identifying the northwest corner of the transportation table. Allocate as much as possible to the cell in the northwest corner, subject to the supply and demand constraints. Move to the next row or column and allocate as much as possible to the first empty cell, subject to the supply and demand constraints. Repeat step 3 until all cells are allocated.

The methodology of Particle Swarm Optimization can be summarized as follows:

Initialization: Initialize the population of particles with random positions and velocities within the search space. Each particle also keeps track of its personal best-known position and the global best-known position of the swarm.

Evaluation: Evaluate the fitness of each particle by calculating the objective function value associated with its current position.

Update personal best-known position: Update the personal best-known position of each particle if its current fitness value is better than its previous best-known fitness value.

Update global best-known position: Update the global best-known position of the swarm by considering the best-known positions of all particles.

Update particle velocities and positions: Update the velocity of each particle based on its previous velocity, the distance to its personal best-known position, and the distance to the global best-known position. Then update the position of each particle based on its updated velocity.

Termination criteria: Check termination criteria to determine whether to stop the optimization process. Termination criteria can be based on the number of iterations, a predefined fitness threshold, or a combination of factors.

Repeat steps 2-6 until the termination criteria are met.

**CODING AND IMPLEMENTATION**

**Transportation North-west :**

def northwest\_corner(costs, supply, demand):

plan = [[0 for j in range(len(demand))] for i in range(len(supply))]

i = 0

j = 0

while i < len(supply) and j < len(demand):

amount = min(supply[i], demand[j])

plan[i][j] = amount

supply[i] -= amount

demand[j] -= amount

if supply[i] == 0:

i += 1

if demand[j] == 0:

j += 1

cost = sum([sum([plan[i][j] \* costs[i][j] for j in range(len(demand))]) for i in range(len(supply))])

return plan, cost

costs = [[2, 3, 1], [5, 4, 8], [5, 6, 8]]

supply = [20, 30, 10]

demand = [25, 15, 20]

plan, cost = northwest\_corner(costs, supply, demand)

print(plan)

print(cost)

**particle swarm optimization:**

import random

import numpy as np

class Particle:

def \_\_init\_\_(self, x0):

self.position = x0

self.velocity = np.zeros\_like(x0)

self.best\_position = x0

self.best\_fitness = float('inf')

def update\_position(self):

self.position += self.velocity

def update\_velocity(self, w, c1, c2, global\_best\_position):

r1 = np.random.rand(len(self.position))

r2 = np.random.rand(len(self.position))

cognitive = c1 \* r1 \* (self.best\_position - self.position)

social = c2 \* r2 \* (global\_best\_position - self.position)

self.velocity = w \* self.velocity + cognitive + social

class PSO:

def \_\_init\_\_(self, fitness\_function, num\_particles, dim, max\_iter, lb, ub, w=0.5, c1=2, c2=2):

self.fitness\_function = fitness\_function

self.num\_particles = num\_particles

self.dim = dim

self.max\_iter = max\_iter

self.lb = lb

self.ub = ub

self.w = w

self.c1 = c1

self.c2 = c2

self.global\_best\_position = None

self.global\_best\_fitness = float('inf')

self.particles = []

for i in range(num\_particles):

x0 = np.random.uniform(lb, ub, dim)

particle = Particle(x0)

self.particles.append(particle)

def optimize(self):

for i in range(self.max\_iter):

for particle in self.particles:

fitness = self.fitness\_function(particle.position)

if fitness < particle.best\_fitness:

particle.best\_position = particle.position

particle.best\_fitness = fitness

if fitness < self.global\_best\_fitness:

self.global\_best\_position = particle.position

self.global\_best\_fitness = fitness

for particle in self.particles:

particle.update\_velocity(self.w, self.c1, self.c2, self.global\_best\_position)

particle.update\_position()

particle.position = np.clip(particle.position, self.lb, self.ub)

return self.global\_best\_position, self.global\_best\_fitness

def rastrigin(x):

A = 10

n = len(x)

return A \* n + np.sum(x\*\*2 - A \* np.cos(2 \* np.pi \* x))

num\_particles = 50

dim = 5

max\_iter = 100

lb = -5.12

ub = 5.12

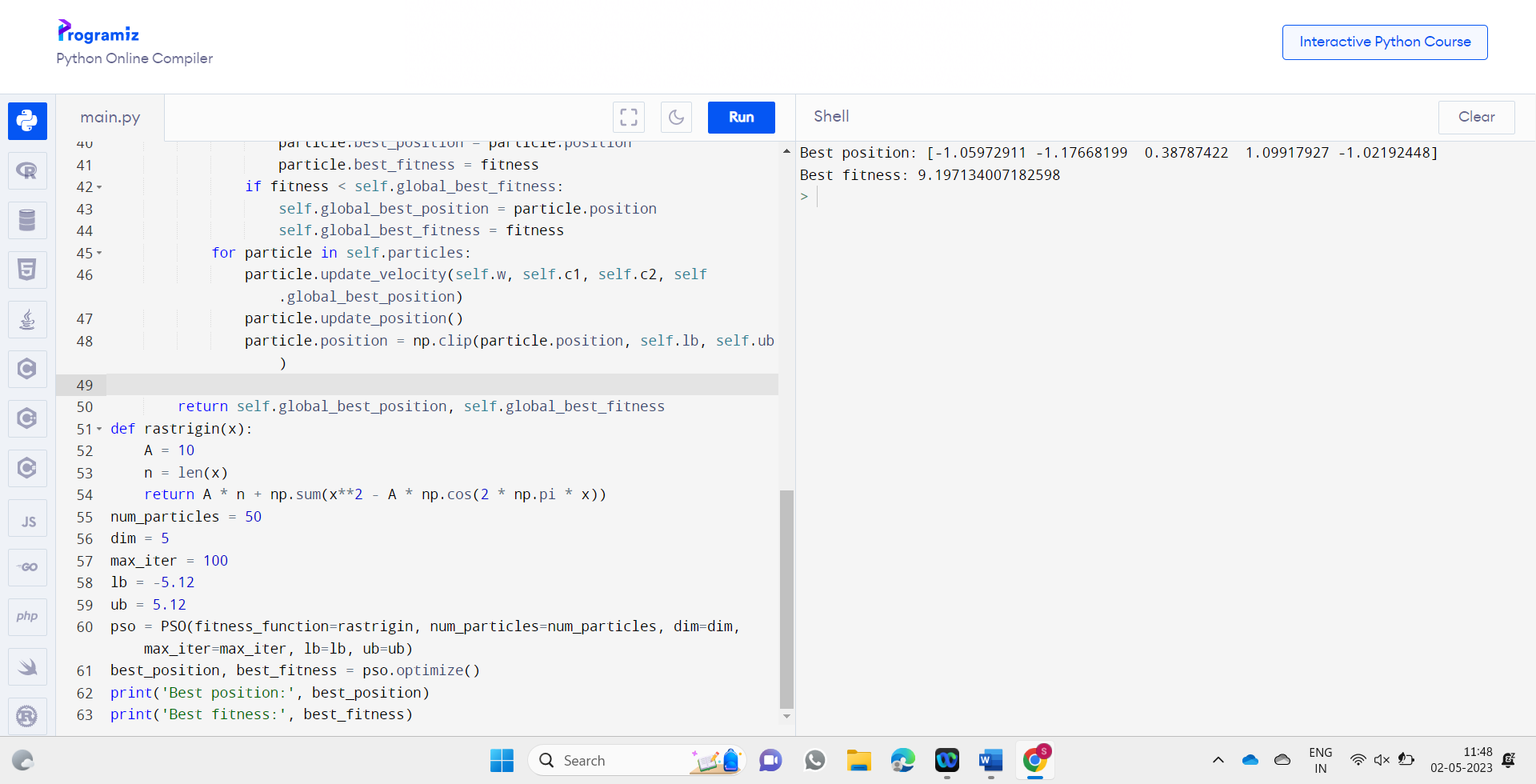
pso = PSO(fitness\_function=rastrigin, num\_particles=num\_particles, dim=dim, max\_iter=max\_iter, lb=lb, ub=ub)

best\_position, best\_fitness = pso.optimize()

print('Best position:', best\_position)

print('Best fitness:', best\_fitness)

**RESULT ANALYSIS**

****

**CONCLUSION**

In conclusion, the North West Corner Method is a useful approach for finding an initial feasible solution to the transportation problem in mathematical programming. It serves as a starting point for subsequent optimization techniques to determine the optimal allocation and minimize the transportation cost.

In conclusion, Particle Swarm Optimization is a powerful optimization algorithm that mimics the collective intelligence of a swarm of particles. By leveraging the collective knowledge of the swarm, PSO efficiently explores the search space and converges towards the optimal solution. Its wide applicability and effectiveness make it a valuable tool in solving various optimization problems.

**FUTURE ENHANCEMENT**

Improved Initial Solution: The North-West Corner Method provides an initial feasible solution, but it may not always be the most optimal. Future enhancements can focus on developing more sophisticated initialization techniques that consider other factors such as cost, demand, and supply constraints to generate better initial solutions.

Advanced Allocation Techniques: The North-West Corner Method follows a sequential allocation approach, where the cells are filled one by one in a fixed order. Future enhancements can explore alternative allocation strategies, such as random allocation or intelligent allocation based on heuristics or optimization algorithms, to improve the quality of the solution.

Variant Designs: PSO has several variant designs, such as constriction factor PSO, adaptive PSO, and quantum-behaved PSO. Future enhancements can explore the development of new variant designs or hybrid approaches that combine PSO with other optimization algorithms to enhance its exploration and exploitation capabilities.

Parameter Adaptation: PSO relies on various parameters, such as the cognitive and social learning factors, inertia weight, and swarm size. Future enhancements can focus on developing adaptive mechanisms or self-adjusting strategies to dynamically adapt these parameters during the optimization process, leading to improved convergence and solution quality.

**BIBLIOGRAPHY**

Bertsimas, D., & Tsitsiklis, J. N. (1997). Introduction to Linear Optimization. Athena Scientific.

Chvatal, V. (1983). Linear Programming. W. H. Freeman.

Dantzig, G. B., Thapa, M. N., & Ramakrishnan, S. (1997). Linear Programming 2: Theory and Extensions. Springer.

Hillier, F. S., & Lieberman, G. J. (2013). Introduction to Operations Research. McGraw-Hill.

Murty, K. G. (1983). Linear Programming. Wiley.

Nocedal, J., & Wright, S. J. (2006). Numerical Optimization. Springer.

Papadimitriou, C. H., & Steiglitz, K. (1998). Combinatorial Optimization: Algorithms and Complexity. Dover Publications.

Schrijver, A. (2003). Combinatorial Optimization: Polyhedra and Efficiency. Springer.

Vanderbei, R. J. (2014). Linear Programming: Foundations and Extensions. Springer.

Wolsey, L. A. (1998). Integer Programming. Wiley.