An Adapted Cat and Mouse Based Optimizer for Solving the Capacitated Vehicle Routing Problem

Duvvuri Prathamesh
Department of Mining Engineering
Indian Institute of Technology (BHU)
Varanasi, India
duvvuri.prathamesh.min23@itbhu.ac.in

Neeraj Ramprasad
ARTPARK
Indian Institute of Science
Bangalore, India
neeraj@artpark.in

Dr. S. N. Omkar
Department of Aerospace Engineering
Indian Institute of Science
Bangalore, India
omkar@iisc.ac.in

***Abstract*—An Adapted Cat and Mouse Based Optimizer (ACMBO) algorithm was proposed for the Capacitated Vehicle Routing Problem (CVRP), a critical combinatorial optimization challenge in logistics often addressed by metaheuristics. ACMBO employs a discrete solution representation, leveraging customer priorities and vehicle reference points, paired with a decoding mechanism to ensure route feasibility and capacity adherence. To enhance exploration and mitigate premature convergence, dynamic customer segmentation via K-means clustering adaptively partitions the search space. The algorithm integrates local search operators and a destroy-and-repair mechanism to refine solutions and escape local optima. When tested on the benchmark cases from CVRPLIB, ACMBO yielded consistent results, matching or providing better solutions than the best-known solutions. When compared with Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), ACMBO consistently demonstrated superior performance, offering a novel approach for solving CVRP effectively across various problem scales.**

***Index Terms*—Capacitated Vehicle Routing Problem, Adapted Cat and Mouse Based Optimizer, Metaheuristics, Optimization**

I. Introduction

The Capacitated Vehicle Routing Problem (CVRP), first introduced by Dantzig and Ramser in 1959 [1], is a cornerstone of combinatorial optimization with significant applications in transportation, logistics, and distribution. As an NP-hard extension of the Traveling Salesman Problem (TSP) [2], CVRP aims to design minimum-cost delivery routes for a fleet of vehicles, starting and ending at a depot, to serve geographically dispersed customers with known demands, while adhering to vehicle capacity constraints. The computational complexity of solving large-scale CVRP instances with exact algorithms has driven the development of heuristic and metaheuristic approaches, which offer near-optimal solutions efficiently. These methods are critical for real-world applications like goods distribution, waste collection, and urban logistics, where cost optimization is paramount.

Among metaheuristic strategies, population-based algorithms inspired by natural processes have shown particular promise due to their ability to explore complex solution spaces and avoid local optima. Notable examples include Particle Swarm Optimization (PSO), adapted for CVRP by Ai and Kachitvichyanukul (2009) [4], Ant Colony Optimization (ACO), enhanced by Stützle and Hoos (2000) [5], and hybrid approaches like the Whale Optimization Algorithm with Grey Wolf Optimizer (WOA-GWO) [8], which achieved strong results on benchmarks. Other innovations include the Firefly Algorithm with local search and genetic operators [11], improved Artificial Bee Colony [10], and adaptive genetic Grey Wolf Optimization [9], all designed to balance exploration and exploitation while addressing challenges like premature convergence.

This study introduces an Adapted Cat and Mouse Based Optimizer (ACMBO), building on the original Cat and Mouse Based Optimizer (CMBO) [3], a population-based metaheuristic inspired by predator-prey dynamics between cats and mice. While CMBO excels in continuous optimization, its application to CVRP's discrete, combinatorial nature requires significant modifications. ACMBO addresses this by introducing a discrete solution representation that encodes feasible routes using customer priorities and vehicle reference points, paired with a feasibility-ensuring decoding mechanism to respect capacity constraints. To enhance exploration, dynamic customer segmentation adaptively clusters customers, while an elite archive retains high-quality solutions to guide the search. Problem-specific local search operators, such as 3-opt for intra-route optimization and inter-route exchanges, refine solutions, and a destroy-and-repair mechanism helps escape local optima. These enhancements make ACMBO a robust and competitive approach for tackling CVRP's complex challenges.

II. Problem Description and Mathematical Modeling

The Capacitated Vehicle Routing Problem (CVRP), an NP-hard extension of the Vehicle Routing Problem (VRP), aims to minimize total travel distance for a fleet of m identical vehicles, each with capacity Q, serving n customers from a single depot (node 0) while satisfying demand constraints. Each customer is visited exactly once by one vehicle, and the total demand per route must not exceed Q. CVRP is modeled on a complete graph $G = (V, E)$, where $V = \{0, 1, \dots, n\}$ represents nodes (depot and customers), and $E = \{(i, j) : i, j \in V, i \neq j\}$ denotes edges with costs $c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Customer demands are given by $q_i : V \to \mathbb{Z}^+, q_0 = 0$, and vehicles are denoted by $K = \{k_1, \dots, k_m\}$.

Assumptions

(i) Known customer demands: $0 < q_i \leq Q$.
(ii) Limited vehicle capacity.

Variables and Parameters

- $V = \{v_0, v_1, \ldots, v_n\}$: Nodes, with v_0 as the depot.
- c_{ij}: Travel cost from node i to j.
- Q: Vehicle capacity.
- m: Number of vehicles.
- $K = \{k_1, \ldots, k_m\}$: Vehicle set.

The CVRP model, introduced by Dantzig and Ramser [1], is defined as:

$$x_{ij}^k = \begin{cases} 1 & \text{if vehicle } k \text{ travels from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

Minimize:

$$Z = \textstyle\sum_{i=0}^{n} \sum_{j=0, j\neq i}^{n} \sum_{k=1}^{m} c_{ij} x_{ij}^k \qquad \text{(i)}$$

Subject to:

$$\textstyle\sum_{k=1}^{m} \sum_{j=1}^{n} x_{ij}^k \leq m, \quad i = 0 \qquad \text{(ii)}$$
$$\textstyle\sum_{i=0}^{n} x_{ij}^k - \sum_{j=0}^{n} x_{ji}^k = 0, \quad k \in K \qquad \text{(iii)}$$
$$\textstyle\sum_{k=1}^{m} \sum_{i=0, i\neq j}^{n} x_{ij}^k = 1, \quad j \in \{1, \ldots, n\} \qquad \text{(iv)}$$
$$\textstyle\sum_{k=1}^{m} \sum_{j=0, j\neq i}^{n} x_{ij}^k = 1, \quad i \in \{1, \ldots, n\} \qquad \text{(v)}$$
$$\textstyle\sum_{j=1}^{n} x_{0j}^k \leq 1, \quad \forall k \in K \qquad \text{(vi)}$$
$$\textstyle\sum_{i=1}^{n} x_{i0}^k \leq 1, \quad \forall k \in K \qquad \text{(vii)}$$
$$\textstyle\sum_{j=1}^{n} \sum_{i=0, i\neq j}^{n} q_j x_{ij}^k \leq Q, \quad \forall k \in K \qquad \text{(viii)}$$
$$\textstyle\sum_{k=1}^{m} \sum_{i\in S} \sum_{j \in S, i\neq j} x_{ij}^k \leq |S| - 1, \quad \forall S \subset V \setminus \{0\} \qquad \text{(ix)}$$

Equation (i) minimizes total distance, subject to constraints (ii)–(viii). Constraint (iii) ensures flow conservation, (iv) and (v) guarantee each customer is visited once, (vi) and (vii) enforce depot start/end, (viii) limits route demand to vehicle capacity, and (ix) eliminates sub-tours.

III. Proposed Adapted Cat and Mouse Based Optimizer

A. Cat and Mouse Based Optimization Algorithm

This section outlines the theoretical framework and mathematical model of the Cat and Mouse-Based Optimization Algorithm (CMBO) [3], designed for optimizing various problems. CMBO, a metaheuristic inspired by cat-mouse dynamics, divides search agents into cats (exploiters) and mice (explorers). These agents navigate the problem's search space through random movements, updated in two phases: cats moving toward mice and mice fleeing to randomly generated havens.

Mathematically, each population member represents a solution, encoded as a vector of problem variables within a population matrix X of size $N \times m$, where N is the number of agents and m is the number of variables [3]. The objective function values are stored in a vector F, and the population is sorted by these values, yielding a sorted matrix X^S and vector F^S. The population is split equally into mice (top half with better objective values) and cats (bottom half), defined as matrices M and C, respectively. In the first phase, cat positions are updated to move toward randomly selected mice using:

$$c_{j,d}^{\text{new}} = c_{j,d} + r \times (m_{k,d} - I \times c_{j,d}), \qquad (1)$$

where $r \in [0, 1]$, $I = \text{round}(1 + \text{rand})$, and the new position is accepted if it improves the objective function. In the second phase, mice escape to havens (randomly chosen positions from the population), updated as:

$$m_{i,d}^{\text{new}} = m_{i,d} + r \times (h_{i,d} - I \times m_{i,d}) \times \text{sign}(F_i^m - F_i^H), \qquad (2)$$

with acceptance based on objective function improvement. Iterations continue until a stopping criterion, such as a fixed number of iterations or acceptable error, is met, yielding a quasi-optimal solution.

B. Initialization of Population

In ACMBO for CVRP, population initialization creates a diverse set of candidate solutions, each encoding vehicle routes to explore the solution space effectively. The population size, $n_{\text{population}}$, balances diversity and efficiency, defined as:

$$n_{\text{population}} = \max\left(50, \min\left(100, 2 \times n_{\text{customers}}\right)\right) \qquad (3)$$

This ensures at least 50 solutions for diversity, capped at 100 for computational efficiency, scaling with the number of customers, $n_{\text{customers}}$. Each solution is a vector of real numbers in $[0, 1]$, with dimensionality:

$$n_{\text{variables}} = n_{\text{customers}} + 2 \times n_{\text{routes}} \qquad (4)$$

where n_{routes} is the number of vehicles. The vector has two parts:

Customer Priorities: The first $n_{\text{customers}}$ elements set customer priorities, determining route construction order.

Vehicle Reference Points: The next $2 \times n_{\text{routes}}$ elements are coordinates guiding customer-to-vehicle assignments by proximity.

This randomized initialization fosters diversity, enabling ACMBO to explore a broad range of feasible CVRP solutions.

C. Customer Clustering

Following population initialization, ACMBO clusters customers spatially to improve CVRP search efficiency. The number of clusters, K, is heuristically determined as:

$$K = \lceil \sqrt{n} \rceil \qquad (5)$$

where n is the number of customers. This choice balances cluster granularity with computational efficiency, ensuring effective segmentation without excessive fragmentation.

The clustering process employs the K-Means algorithm [14], which minimizes the within-cluster sum of squared distances, defined as:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|^2 \qquad (6)$$

where $x_i \in \mathbb{R}^2$ denotes the spatial coordinates of customer i, $\mu_k \in \mathbb{R}^2$ is the centroid of cluster k, and r_{ik} is a binary indicator of cluster membership (1 if customer i is assigned to cluster k, 0 otherwise). The algorithm iteratively assigns each customer to the cluster with the nearest centroid, based on Euclidean distance, and updates centroids as the mean coordinates of assigned customers, converging to a local minimum of J. Cluster assignments are stored as lists of customer indices for each cluster. This clustering mechanism by grouping customers into spatially proximate subsets, focuses search efforts on constructing efficient routes within each cluster, significantly reducing the combinatorial complexity of the CVRP.

1) Dynamic Adaptability: To enhance adaptability in the ACMBO for the CVRP, the algorithm dynamically adjusts customer cluster assignments to reflect search progress, thereby improving exploration and convergence. Every i_{cluster} iterations, after the initial n_{cluster} iterations, the algorithm evaluates the relative improvement in the global best fitness over the preceding i_{cluster} iterations:

$$\Delta f_t = \frac{f_{t-50} - f_t}{f_{t-50}}, \quad f_{t-50} > 0 \tag{7}$$

where f_t is the current global best fitness, and f_{t-50} is the fitness 50 iterations prior. If the improvement (Δf_t) is insufficient, indicating potential stagnation, the number of clusters, K_t, is adjusted within defined bounds:

$$K_t = \max\left(2, \min\left(\lceil \sqrt{n} \rceil, K_{\text{base}} + \lfloor 5(1 - \Delta f_t) \rfloor\right)\right) \tag{8}$$

where n is the number of customers, and $K_{\text{base}} = \max(2, \lfloor \sqrt{n}/2 \rfloor)$ establishes a baseline cluster count. Customers are then re-partitioned using the K-Means algorithm, minimizing the within-cluster sum of squared distances based on their spatial coordinates. The updated cluster assignments are used to reassign search agents (cats) to clusters, ensuring that optimization efforts align with the evolving solution landscape. This dynamic reconfiguration mitigates the risk of local optima entrapment, enabling the algorithm to explore alternative customer groupings that enhance route efficiency while maintaining adherence to capacity constraints.

These clusters guide the assignment of search agents (cats) to optimize sub-routes within localized customer groups, leveraging spatial coherence to minimize route distances while ensuring adherence to vehicle capacity constraints, thereby facilitating effective solution refinement in subsequent optimization phases.

D. Fitness Evaluation

Following customer clustering, the ACMBO evaluates the fitness of each solution for the CVRP by summing route distances plus a 10,000 units penalty per unserved customer. The mathematical expression for evaluating fitness:

$$\sum_{\text{route}\in\text{routes}} \left[d(0, \text{route}[0]) + \textstyle\sum_{i=1}^{|\text{route}|-1} d(\text{route}[i-1], \text{route}[i]) \right.$$
$$\left. + \, d(\text{route}[-1], 0) \right]$$
$$+ \, 10000 \cdot (n_{\text{customers}} - |\text{assigned}|)$$

where $d(a, b)$ denotes the Euclidean distance between locations a and b, with 0 representing the depot, route is a sequence of customers assigned to a vehicle, $n_{\text{customers}}$ is the total number of customers, and $|\text{assigned}|$ is the number of customers served in the solution.

E. Population Sorting

In each ACMBO iteration for CVRP, the $n_{\text{population}}$ solutions are sorted by fitness (total route distance plus penalties for unserved customers). The top half are labeled *mice* (high-quality solutions), and the bottom half are *cats* (solutions needing improvement). The $n_{\text{cats}} = n_{\text{population}}/2$ cats are distributed across K_t active clusters, with roughly n_{cats}/K_t cats per cluster. Remaining cats are assigned to earlier clusters for balance. This enables cats to optimize cluster-specific customer subsets, refining routes locally while preserving global solution diversity.

1) Fitness Sharing: To enhance diversity within clusters, ACMBO's fitness sharing penalizes similar cat solutions. For a cat solution i, its fitness f_i is adjusted as:

$$f_i' = \frac{f_i}{\text{niche}_\text{count}_i} \cdot (1 - \text{crowding\_factor}_i) \tag{9}$$

where $\text{niche}_\text{count}_i = 1 + \sum_{j \neq i} \exp\left(-\frac{\|s_i - s_j\|_I^2}{2\sigma^2}\right)$ measures similarity between solution vectors s_i and s_j, using squared Euclidean distance over cluster-specific customer indices I. The parameter $\sigma = 0.5\sqrt{|I|}$, with $|I|$ as the number of cluster customers, scales similarity. The crowding_factor_i, based on crowding distance, reflects solution density around s_i, normalized to favor dispersed solutions. This reduces fitness for similar solutions, promotes diverse subroute configurations, enhances exploration, and prevents premature convergence in CVRP.

F. Solution Update and Integration

This phase encompasses updating lower-quality solutions (*cats*), aggregating and integrating them with high-quality solutions (*mice*), and further refining the mice to enhance overall solution quality. These steps, executed in each iteration, leverage cluster-specific optimizations and maintain population diversity, driving convergence toward an optimal set of routes that minimizes distance to be covered while adhering to capacity conditions of CVRP [1].

1) Update Cats: In each iteration of ACMBO, cat solutions, assigned to specific clusters, are updated to optimize subroutes for their respective customers. The update modifies the solution vector's components corresponding to the cluster's customers (indices I) and vehicle reference points:

$$s_{\text{new}}[I] = s_{\text{cat}}[I] + r \cdot (\text{mouse}[I] - I_{\text{rand}} \cdot s_{\text{cat}}[I]) \cdot (1 + \alpha \cdot t / n_{\text{iterations}}) \tag{10}$$

where s_{cat} is the current cat solution, mouse is a randomly selected high-quality solution, $r \in [0, 1]$ is a random factor, $I_{\text{rand}} \in \{1, 2\}$ introduces variability, $\alpha = 0.5$ scales exploration, and $t/n_{\text{iterations}}$ adjusts exploration intensity over time. A small probability of random component swaps enhances diversity, and a temperature-based criterion occasionally accepts worse solutions to promote exploration. This focused update ensures cat solutions refine cluster-specific subroutes, contributing to efficient route configurations in the CVRP solution space.

2) Solution Integration: Updated cat solutions from each cluster are aggregated into a unified cat population, which is then combined with the mice to create new population for the further iteration. Let S_k denote the set of updated cat solutions for cluster k, with $|S_k|$ solutions proportional to the number of cats assigned to the cluster. The cat population is formed as:

$$S_{\text{cats}} = \bigcup_{k=1}^{K_t} S_k \tag{11}$$

where K_t is the number of active clusters. The new population is constructed by:

$$S = S_{\text{mice}} \cup S_{\text{cats}} \tag{12}$$

where S_{mice} represents the current mice solutions. This integration preserves cluster-specific optimizations, as each cat's solution retains improved subroutes for its assigned customers, while the inclusion of mice ensures high-quality solutions are maintained. The diversity introduced by fitness sharing in the previous step ensures that aggregated cat solutions propose varied subroute configurations, enhancing global exploration.

3) Update Mice: To refine high-quality solutions, each mouse solution is updated:

$$s_{\text{new}} = s_{\text{mouse}} + r \cdot (\text{pop}[l] - I_{\text{rand}} \cdot s_{\text{mouse}}) \cdot \text{sign} \tag{13}$$

where $\text{pop}[l]$ is a randomly selected solution from the population, $r \in [0, 1]$, $I_{\text{rand}} \in \{1, 2\}$, and $\text{sign} \in \{-1, 0, 1\}$ depend on the fitness comparison between the mouse and the selected solution. Mutations, applied with probability p_m, introduce random swaps to prevent stagnation. Updates are accepted only if they improve fitness, ensuring that mice remain the best solutions in the population, guiding subsequent cat updates and driving convergence toward optimal CVRP routes.

G. Archiving, Iteration Transition, and Termination

In ACMBO for CVRP, each iteration concludes by tracking the best solution, archiving high-quality solutions, and transitioning to the next iteration to refine the population, ensuring convergence to optimal routes that minimize distance while satisfying capacity constraints. The algorithm terminates after $n_{\text{iterations}}$ iterations.

Tracking and Archiving: The $n_{\text{population}}$ solutions are evaluated to identify the lowest fitness (route distance plus penalties for unserved customers). If below the global best, f_{global}, the global best solution, s_{global}, its routes, and stagnation counter, $n_{\text{stagnation}}$, are updated, resetting the counter. This solution is stored in an elite archive, limited to $\max(5, \lfloor n/10 \rfloor)$ solutions (where n is customer count), sorted by fitness, with excess discarded. The global best fitness is logged in a 50-entry history for dynamic clustering. If no improvement occurs, $n_{\text{stagnation}}$ increments, signaling stagnation.

Iteration Transition: The updated population of mice and cats advances to the next iteration, leveraging cluster-specific optimizations to continue refining routes via clustering, sorting, fitness sharing, and updates.

These steps preserve optimal solutions, monitor progress, and enable population refinement, guiding ACMBO toward an optimal CVRP route configuration until termination after $n_{\text{iterations}}$.

H. Local Search Operators

ACMBO employs local search mechanisms, Variable Neighborhood Descent (VND) and Destroy-and-Repair, to refine CVRP solutions, improving route quality by perturbing and optimizing to escape local optima and enhance convergence.

1) Variable Neighborhood Descent (VND): The `variable_neighborhood_descent` function in ACMBO refines CVRP routes using six operators, accepting changes that reduce total distance while ensuring feasibility within capacity constraints: Intra-Insert relocates a customer within a route to minimize distance; Intra-Change swaps two customers to shorten the route; Intra-Reverse reverses a route segment for better connectivity; 3-Opt tests three-edge exchanges (for routes with ≥ 4 customers) to reduce distance [7]; Inter-Insert transfers a customer to another route at the least costly position; and Inter-Change swaps customers between routes if feasible and distance decreases.

2) Destroy-and-Repair: Triggered by a stagnation counter, Destroy-and-Repair perturbs global best routes to boost exploration, using `destroy_solution` and `repair_solution`. The *destroy phase* removes up to $k = \max(5, \lfloor n/10 \rfloor)$ customers (where n is customer count) based on correlation $R_{ij} = 1/(d_{ij}/\max_d + V_{ij} + 10^{-6})$, where d_{ij} is inter-customer distance, $\max_d$ is max pairwise distance, and $V_{ij} = 0$ if customers share a route, else 1, targeting close customers across routes. In contrast, the *repair phase* reinserts removed customers using cheapest insertion, minimizing distance while ensuring capacity. If infeasible, retains original solution. Adjusts routes to match required vehicle count.

IV. Simulation Experiments and Analysis

TABLE I
PARAMETER SETTINGS FOR ACMBO ALGORITHM

Parameters	Value
$n_{\text{iterations}}$	1500
α	0.5
n_{vnd}	40
n_{restart}	100
i_{restart}	5
n_{cluster}	50
i_{cluster}	50
capacity	Per CVRPLIB
No. of vehicles, k	Per CVRPLIB
No. of customers, n	Per CVRPLIB

Algorithm 1 is the pseudo code for the ACMBO algorithm, having all steps of ACMBO algorithm in order. Table 1 details the input parameter settings for the ACMBO algorithm used in simulations on CVRPLIB test cases. The Particle Swarm Optimization (PSO) [4] and Ant Colony Optimization (ACO) algorithms [12] are simulated with identical population sizes, using parameters from their respective referenced literature.

Algorithm 1 Adapted Cat and Mouse Based Optimizer (ACMBO) for CVRP

```
1:  Input: $n_{\text{customers}}$, $n_{\text{routes}}$, $\alpha$, $n_{\text{vnd}}$, $n_{\text{restart}}$, $n_{\text{iterations}}$, $i_{\text{restart}}$, $n_{\text{cluster}}$, $i_{\text{cluster}}$
2:  Output: Best solution $s_{\text{global}}$, fitness $f_{\text{global}}$, routes
3:  Initialize population $S$ with $n_{\text{population}}$ and $n_{\text{variables}}$ (Section III.B, Equations 3, 4)
4:  Cluster customers into $K$ clusters (Section III.C, Equation 5)
5:  for $t = 1$ to $n_{\text{iterations}}$ do
6:      Evaluate fitness (Section III.D)
7:      Sort population, assign mice and cats (Section III.E)
8:      Apply fitness sharing to cats (Section III.E.1, Equation 9)
9:      for each cat solution do
10:         Update cat solution (Section III.F.1, Equation 10)
11:     end for
12:     Integrate solutions $S$ (Section III.F.2, Equations 10, 11)
13:     for each mouse solution do
14:         Update mouse solution (Section III.F.3, Equation 12)
15:     end for
16:     Track best solution, archive (Section III.G)
17:     if $t \geq n_{\text{cluster}}$ and $t \bmod i_{\text{cluster}} = 0$ then
18:         Adjust clusters dynamically (Section III.C.1, Equations 7, 8)
19:     end if
20:     Apply VND (Section III.H.1)
21:     if stagnation then
22:         Apply Destroy-and-Repair (Section III.H.2)
23:     end if
24: end for
25: Return $s_{\text{global}}$, $f_{\text{global}}$, routes
```

A. Implementation of ACMBO and Comparison with other Metaheuristic Algorithms

ACMBO was tested on the benchmark instances of CVRPLIB using Python 3.13 in VS Code on a Windows 11 PC (Intel Core i7-10510U, 8 GB RAM). Performance was measured over 10 runs via best (lowest objective value), worst (highest), and average values. Solution quality was assessed by percentage deviation (Dev.%) from best-known solutions (BKS) in CVRPLIB, per Equation (14):

$$\text{Dev.\%} = \frac{S - \text{BKS}}{\text{BKS}} \times 100 \tag{14}$$

Here, S is the best solution obtained, and BKS is the CVRPLIB BKS. A 0% gap indicates an optimal solution, near-zero suggests a good solution, and larger gaps indicate poorer solutions.

Table II shows ACMBO results across 35 CVRP instances from Augerat's Sets A, B, P, Christofides and Eilon's Set E, Fisher's Set F, Christofides, Mingozzi, and Toth's Set M [6], and Rochat and Taillard's Set Tai taken from CVRPLIB. ACMBO's results across 35 CVRP instances (Table II) show high precision, achieving optimal BKS in seven instances (0.00% deviation) and surpassing BKS in two (-2.49%, -4.87%).

In Table III, ACMBO's performance was compared to that of PSO and ACO. Bold values in the table denote the best results, with (*) for optimal BKS matches and (**) for

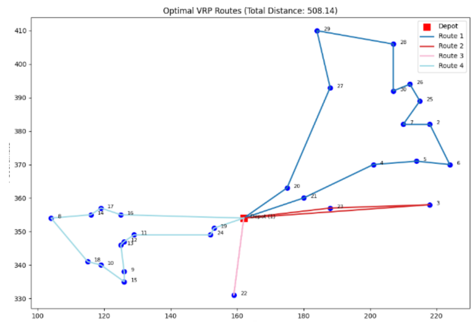

Fig. 1. Best solution obtained for the E-n30-k4 test case

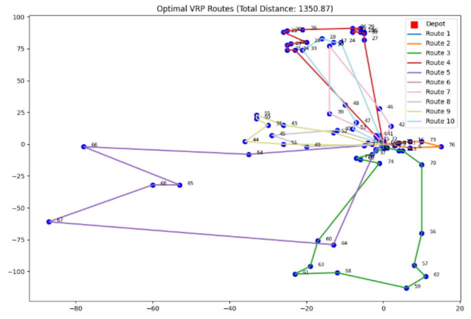

Fig. 2. Best solution obtained for the Tai75b test case

TABLE II
ACMBO RESULTS WITH DEVIATION PERCENTAGE

Instance	BKS	Min.	Avg.	Max.	Dev. (%)
P-n16-k8	451	451	451	451	0.00
P-n19-k2	212	212	212	212	0.00
P-n20-k2	216	216	216	216	0.00
P-n21-k2	211	211	211	211	0.00
P-n22-k8	603	588	588.6	591	-2.49
P-n23-k8	529	531	531.4	533	0.38
A-n32-k5	784	787	789	790	0.38
A-n33-k5	661	662	669	713	0.15
A-n34-k5	778	786	790	793	1.03
A-n36-k5	799	802	806.8	815	0.37
A-n37-k5	669	678	681.21	687	1.34
A-n44-k6	937	946	964	982	0.96
A-n45-k7	1146	1153	1185	1215	0.61
A-n54-k7	1167	1176	1191	1207	0.77
A-n62-k8	1288	1299	1319.8	1333	0.85
E-n22-k4	375	375	378	381	0.00
E-n23-k3	569	569	569	569	0.00
E-n30-k4	534	508	518.4	534	-4.87
E-n33-k4	835	837	841.25	848	0.24
E-n51-k5	521	524	528.5	533	0.57
B-n31-k5	672	679	683.3	685	1.04
B-n38-k6	805	807	810.4	817	0.25
B-n44-k5	909	914	917	921	0.55
B-n45-k6	678	691	701	707	1.92
B-n51-k7	1032	1036	1038.5	1044	0.39
B-n57-k9	1598	1605	1611.75	1620	0.44
B-n63-k10	1496	1507	1532.7	1552	0.74
B-n67-k10	1032	1057	1068.75	1074	2.42
B-n78-k10	1221	1264	1282.2	1344	3.52
F-n45-k4	724	728	735.8	749	0.55
F-n72-k4	237	241	243.4	247	1.69
Tai75a	1618.36	1630.2	1636.4	1653	0.73
Tai75b	1344.62	1350.87	1361	1389	0.46
Tai100c	1406.2	1423.9	1437	1451	1.26
Tai150b	2727.03	2755	2784.7	2821	1.03

solutions surpassing BKS. ACMBO outperforms or matches PSO and ACO in 29 of 35 instances (83%), consistently yielding lower objective values than ACO (differences up to 442.77 in Tai75a) and outperforming PSO in 32 instances, with PSO better in three (by 3–5 units). ACMBO's low deviations across small (16 customers) to large (150 customers) instances highlight its scalability and effectiveness, offering superior precision and consistency over PSO and ACO.

The plots in Fig 1. and Fig 2. illustrate the best solutions achieved by the ACMBO algorithm when applied to the E-n30-k4 and Tai75b test cases, respectively. Specifically, Fig 2. highlights the ACMBO algorithm's performance when evaluated on an instance featuring a non-uniform, clustered arrangement of customers, thereby demonstrating its effectiveness for a diverse distribution of customers.

TABLE III
Comparative Performance on CVRP

Instance	BKS	ACMBO	ACO	PSO	Dev.(%)
P-n16-k8	451	**451***	451	451	0.00
P-n19-k2	212	**212***	212	212	0.00
P-n20-k2	216	**216***	216	216	0.00
P-n21-k2	211	**211***	211	211	0.00
P-n22-k8	603	**588****	625	609	-2.49
P-n23-k8	529	**531**	535	531	0.38
A-n32-k5	784	**787**	794	787	0.38
A-n33-k5	661	**662**	680	662	0.15
A-n34-k5	778	**786**	801	786	1.03
A-n36-k5	799	**802**	811	802	0.37
A-n37-k5	669	678	695	**672**	1.34
A-n44-k6	937	**946**	972	952	0.96
A-n45-k7	1146	**1153**	1208	1167	0.61
A-n54-k7	1167	**1176**	1208	1189	0.77
A-n62-k8	1288	**1299**	1371	1329	0.85
E-n22-k4	375	**375***	414	379	0.00
E-n23-k3	569	**569***	570	569	0.00
E-n30-k4	534	**508****	553	535.3	-4.87
E-n33-k4	835	**837**	938	846	0.24
E-n51-k5	521	**524**	548	524	0.57
B-n31-k5	672	679	679	**676**	1.04
B-n38-k6	805	**807**	834	807	0.25
B-n44-k5	909	**914**	936	926	0.55
B-n45-k6	678	691	734	**686**	1.92
B-n51-k7	1032	**1036**	1065	1044	0.39
B-n57-k9	1598	**1605**	1645	1616	0.44
B-n63-k10	1496	**1507**	1591	1530	0.74
B-n67-k10	1032	**1057**	1152	1057	2.42
B-n78-k10	1221	**1264**	1313	1264	3.52
F-n45-k4	724	**728**	735.8	749	0.55
F-n72-k4	237	**241**	243.4	247	1.69
Tai75a	1618.36	**1630.2**	1636.4	1653	0.73
Tai75b	1344.62	**1350.87**	1361	1389	0.46
Tai100c	1406.2	**1423.9**	1437	1451	1.26
Tai150b	2727.03	**2755**	2784.7	2821	1.03

V. Conclusion

This paper introduces the Adapted Cat and Mouse-based Optimizer (ACMBO), an enhancement of the Cat and Mouse-based Optimizer [3], designed for effectively solving the Capacitated Vehicle Routing Problem (CVRP). To validate ACMBO's approach for solving CVRP, it was tested on 35 diverse benchmark instances from CVRPLIB. Results showed that ACMBO consistently delivered near-optimal solutions across small to large problem sizes, often achieving or surpassing the best-known solutions (BKS). In addition, ACMBO's results were compared with those of well-established meta-heuristic algorithms, such as PSO and ACO, demonstrating that ACMBO outperformed both algorithms in terms of both precision and accuracy. However, ACMBO did exhibit minor limitations in larger CVRP instances with higher customer counts, where the deviation from BKS reached up to 3.52%, suggesting a potential area for refinement. Future work includes enhancing ACMBO algorithm to solve various variants of the Vehicle Routing Problem (VRP), like the Electric VRP, VRP with time-windows, and multiple-depots, which would test its ability to incorporate a variety of logistical constraints. Furthermore, ACMBO can be used for solving real-world applications like optimizing routes for e-commerce services with real-time data and dynamic constraints which could validate ACMBO's practical applicability.

VI. Acknowledgment

This research has been supported by Computational Intelligence (CINT) Lab, Department of Aerospace Engineering, IISc, Bangalore and IIT (BHU), Varanasi.

References

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, Oct. 1959.

[2] C. Contardo and R. Martinelli, "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints," *Discrete Optimization*, vol. 12, pp. 129–146, May 2014.

[3] M. Dehghani, Š. Hubálovský, and P. Trojovský, "Cat and mouse based optimizer: A new nature-inspired optimization algorithm," *Sensors*, vol. 21, no. 15, p. 5214, Jul. 2021.

[4] Ai, The Jin, and Voratas Kachitvichyanukul. "A Particle Swarm Optimization for the Capacitated Vehicle Routing Problem." *International Journal of Logistics and SCM Systems*, vol. 2, no. 1, 2007, pp. 50–55.

[5] B. Bullnheimer, R. F. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem," in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voß, S. Martello, I. H. Osman, and C. Roucairol, Eds. Boston, MA, USA: Springer, 1999, pp. 285–296.

[6] Z. H. Ahmed et al., "An enhanced ant colony system algorithm based on subpaths for solving the capacitated vehicle routing problem," *Symmetry*, vol. 15, no. 11, p. 2020, Nov. 2023.

[7] A. S. Alfa, S. S. Heragu, and M. Chen, "A 3-opt based simulated annealing algorithm for vehicle routing problems," *Computers & Industrial Engineering*, vol. 21, no. 1–4, pp. 635–639, 1991.

[8] V. H. S. Pham, V. N. Nguyen, and N. T. N. Dang, "Hybrid whale optimization algorithm for enhanced routing of limited capacity vehicles in supply chain management," *Scientific Reports*, vol. 14, no. 1, p. 793, Jan. 2024.

[9] L. Korayem, M. Khorsid, and S. S. Kassem, "Using grey wolf algorithm to solve the capacitated vehicle routing problem," in *IOP Conf. Series: Materials Science and Engineering*, vol. 83, no. 1, 2015, p. 012014.

[10] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European J. Operational Research*, vol. 215, no. 1, pp. 126–135, Nov. 2011.

[11] A. M. Altabeeb, A. M. Mohsen, and A. Ghallab, "An improved hybrid firefly algorithm for capacitated vehicle routing problem," *Applied Soft Computing*, vol. 84, p. 105728, Nov. 2019.

[12] Yu, Bin, Zhong-Zhen Yang, and Baozhen Yao. "An improved ant colony optimization for vehicle routing problem." *European journal of operational research* 196.1 (2009): 171-176.

[13] M. Alssager et al., "Hybrid cuckoo search for the capacitated vehicle routing problem," *Symmetry*, vol. 12, no. 12, p. 2088, Dec. 2020.

[14] M. A. Mohammed et al., "Solving vehicle routing problem by using improved K-nearest neighbor algorithm for best solution," *J. Computational Science*, vol. 21, pp. 232–240, Jul. 2017.