4

Meta-Heuristic Clustering Frameworks for Vehicular Communication Reliability

VANETs face critical routing challenges when optimizing across areas with varying vehicle densities, like congested city intersections versus sparse highways where conventional algorithms either explore too broadly (causing slow convergence) or exploit too narrowly (risking suboptimal local solutions). For example, during rush hour transitions, established vehicle clusters frequently collapse as traffic patterns shift, causing connection failures and excessive rerouting (a disadvantage we were having in direction aware protocols as they relied on static heuristics). After directional routing, we now focus on clustering longevity and multi-constraint routing using meta-heuristic methods. This chapter proposes hybrid nature-inspired approaches (MetaLearn and MRMH) combining population-based algorithms (GWO) for global network exploration with local refinement mechanisms (Temporal Difference Learning (TDL) and SQP) to simultaneously optimize cluster stability and routing paths while dynamically adapting to changing conditions.

The remainder of this chapter is organized as follows. Section 4.1 discusses the concepts of network stability and optimization in VANETs, explaining why they are critical and how clustering contributes to these goals. Section 4.2 introduces the MetaLearn framework, detailing its network model assumptions, the clustering approach using GWO, and the route adaptation using TDL. Section 4.3 presents the MRMH framework, describing its design motivations, the hybrid GWO-SQP methodology, and the enhanced fitness measures for clustering. In Section 4.4, we describe the simulation framework used to evaluate these approaches and provide a performance evaluation, including the rationale for chosen traffic scenarios and metrics. Finally, the chapter's findings are summarized in the context of VANET reliability improvement.

4.1 CLUSTERING AND ROUTE OPTIMIZATION IN VANETS

Network stability ensures that the network can continue to function effectively over time despite frequent and unpredictable topology changes. In practical terms, a stable VANET network maintains ongoing communication links with minimal interruptions even as vehicles move; this consistency is vital for safety messages that depend on continuous coverage. Optimization, meanwhile, refers to enhancing key performance metrics of the network, for example, maximizing data throughput, minimizing end-to-end packet delay, or reducing the overhead (such as control message load) needed to deliver messages [93]. An optimized VANET network makes the most efficient use of its resources (bandwidth, channel capacity, node buffer space) to deliver information quickly and reliably. Both facets are essential: a perfectly stable network that

is very slow or has low throughput would fail to meet application needs, just as a fast network that constantly drops connections would be unsustainable.

In VANETs, stability and optimization are especially challenging because of the high mobility and decentralized nature of the network. Routing optimization is often framed as an objective function to minimize or maximize (e.g., minimize total travel time for a packet, or maximize the probability of delivery within a deadline) subject to the constraints of a dynamic graph. Solving such problems optimally is typically NP-hard, and the problem state (the network topology) is constantly changing. Therefore, a multi-faceted approach is needed, one that can provide good solutions quickly and update them as conditions change. Clustering is one such approach that significantly aids in routing, better channel access management and QoS assurance [115]. By grouping vehicles into clusters (based on geographic proximity or similar movement patterns), the network can localize the effects of mobility. Within a cluster, member vehicles might communicate frequently but only the CH needs to exchange routing information with other clusters. This hierarchy reduces the routing complexity and overhead across the network as a whole. If clusters are formed in a stable way (say, vehicles moving in the same direction or at similar speeds are in the same cluster), the cluster membership remains relatively constant over some time, which improves network stability. Only when a vehicle drastically changes direction or leaves the area does it need to switch clusters, which is less frequent if clustering criteria are chosen well.

Stable clusters thus serve as an anchor for the network, improving scalability and reliability. With clusters in place, routing can be done in two tiers: first within clusters and then between CHs. This confinement means route changes due to mobility often only require local adjustments inside a cluster rather than across the entire network. As a result, even though the VANET topology as a whole is changing, at the level of cluster-to-cluster connectivity the changes are slower, yielding a form of stability. Many studies emphasize cluster stability as a crucial factor because unstable clusters (those that reform too often) cause excessive control signaling and packet loss.

While clustering contributes to stability (network connectivity), we must still optimize the routing within and between clusters. After forming clusters, one needs to select optimal routes for data packets. This can be viewed as finding optimal paths in a graph where vertices are vehicles (or CHs) and edges represent wireless links. Researchers have applied various graph-theoretic models to VANETs [75], often assigning weights to edges based on signal strength, link duration (how long a link is expected to last given relative speed), or estimated transmission delay. The goal is to find paths that minimize delay or maximize reliability. However, because VANET graphs change quickly and have many possible paths, classic shortest-path algorithms are not sufficient, they would need constant re-computation and cannot easily handle multiple criteria (like simultaneously minimizing delay and maximizing link stability).

This is where meta-heuristic techniques come into play as powerful tools for VANET optimization. Meta-heuristics (such as Genetic Algorithms [116], Particle Swarm Optimization [117], Tabu Search [118], etc.) can handle complex objective functions and constraints by exploring the solution space stochastically. They don't guarantee a global optimum, but they can often find a very good solution in a fraction of the time that brute-force search would require. For example, one could encode a candidate clustering or routing plan as a "chromosome" in a genetic algorithm and evolve a population of such chromosomes to optimize network performance metrics. Similarly, swarm algorithms might treat possible next-hop selections as particles moving toward an optimal configuration [81]. These approaches have been recognized as effective for problems like multi-hop routing in MANETs and VANETs.

However, applying meta-heuristics to VANETs is not without difficulties. Computational complexity is a concern: meta-heuristics reduce the search space but can still be slow if the scenario is large (hundreds of vehicles) or if the algorithm's parameters are not well-tuned. More importantly, VANETs' dynamic nature can invalidate a solution quickly. A meta-heuristic might need dozens or hundreds of iterations to converge, and within that time vehicles may have moved significantly. Thus, a solution found at iteration 100 might not be optimal or even feasible by the time it's implemented. This motivates combining fast-reacting mechanisms with the meta-heuristic's planning. For instance, one can use an approach that finds a good clustering, but then continuously adapts routing on the fly as conditions change – which is exactly the philosophy behind our MetaLearn framework. Another strategy is to hybridize meta-heuristics with faster local optimization techniques that can adjust solutions incrementally (which MRMH does by integrating SQP).

In summary, maintaining network stability and route efficiency in VANETs calls for adaptive clustering and optimize routing algorithms. Clustering brings order to the network, reducing volatility by grouping vehicles, while optimization algorithms ensure that the chosen clusters and routes yield high performance (throughput, low delay, etc.). The subsequent sections introduce solutions that embody these principles. MetaLearn and MRMH are designed to stabilize the network by forming intelligent clusters and to optimize performance by dynamically adjusting to the VANET's changing state. They illustrate how combining clustering with meta-heuristic optimization and learning can significantly improve reliability and service quality in vehicular networks.

4.2 METALEARN: OPTIMIZING ROUTING HEURISTICS WITH A HYBRID META-LEARNING APPROACH IN VANETS

Recent advances in VANET routing [119, 120] have explored both evolutionary algorithms (for efficient route search) and machine learning (for adaptive decision-making). MetaLearn is a framework that synergistically combines these two paradigms to enhance routing reliability in clustered VANET environments. The name "MetaLearn" reflects its dual nature: a meta-heuristic component finds a near-optimal clustering and routing structure, and a learning component continuously improves that structure as conditions change. The primary goal of MetaLearn is to ensure that data finds a stable, low-latency path through the network without incurring the heavy recomputation overhead that a purely heuristic or purely reactive approach might entail.

In MetaLearn, we integrate the GWO algorithm with a TDL strategy. GWO is a swarm intelligence meta-heuristic inspired by the cooperative hunting behavior of grey wolves in nature, and we use it to determine efficient cluster formations and initial routing decisions. TDL, on the other hand, is a reinforcement learning approach (closely related to Q-learning) that allows the system to learn from real-time network feedback (such as observed packet delays or link breakages) and adjust routing decisions accordingly. By combining GWO with TDL, MetaLearn leverages the global search capability of the former and the online adaptability of the latter. This is a novel approach in VANET routing; prior works have typically employed meta-heuristics and learning separately. In MetaLearn, the two work in combination: GWO provides a strong starting solution, and the learning algorithm refines that solution continuously, which helps avoid the need to restart the heuristic search every time a vehicle moves or traffic conditions change.

From a high-level perspective, MetaLearn operates in two phases: (1) an optimization phase where clustering is optimized using GWO, and (2) an adaptation phase where the routing is fine-tuned using TDL as the vehicles move and network state evolves. Importantly, these phases are not strictly sequential one-time events, MetaLearn can intermittently invoke the optimization phase again if the network conditions have drastically changed (to re-optimize clusters), but on a shorter time scale, the adaptation phase is continuously running to tweak routes. This design ensures that MetaLearn can react to quick changes (through learning) and also periodically recalibrate the overall structure (through heuristic optimization) when needed.

4.2.1 Problem Definition and Preliminaries

We represent the VANET as a graph $G = (V, E)$ where V is the set of vehicles (nodes) and E is the set of wireless communication links (edges) between vehicles. An edge $(u, v) \in E$ exists if and only if vehicles u and v are within wireless range of each other and can communicate directly. In general, VANET links are considered bidirectional (symmetric radio range), but it is straightforward to treat G as a directed graph if links are asymmetric (the MetaLearn formulation does not depend on symmetry). At any given time, G captures the connectivity of the network. As vehicles move, the set E changes over time (edges appear or disappear), so we can think of $G(t)$ as the graph at time t. We also include a set $R = \{r_1, r_2, \ldots, r_p\}$ of RSU in the model, which act as stationary nodes that vehicles can also communicate with. RSUs can provide internet gateway functionality or simply extend the range of communications. If a vehicle is within range of an RSU, it can use the RSU to indirectly communicate with other vehicles that are also connected to an RSU (multi-hop V2I2V communication). This means our graph model can have edges representing V2V links (vehicle-to-vehicle) or V2I links (vehicle-to-infrastructure). We assume, consistent with real-world deployments, that each vehicle is equipped with an OBU supporting DSRC radios. DSRC (IEEE 802.11p) offers direct vehicle communication in the 5.9 GHz band with an effective range on the order of 100-300 meters under normal conditions. This range is reflected in our graph E: an edge exists if two vehicles are within 300m (line of sight). Vehicles periodically broadcast beacon messages (hello packets) to announce their presence and recent position/velocity, which is a standard practice in VANET protocols and is assumed in our model for neighbor discovery.

The spatial scope of our model is a regional network of vehicles, for example, a few kilometers of an urban arterial road network or a segment of highway, where vehicles within that area can potentially form a multi-hop network. We choose an urban scenario (Jodhpur, India) as a representative environment in our simulations, but the model itself is general. The temporal scope for clustering decisions is on the order of seconds: we aim to maintain clusters that last for several beacon intervals (each beacon interval is 100 ms, depending on configuration), adapting only as needed. This aligns with practical VANET clustering proposals which try to avoid cluster reconfigurations happening too often (to prevent overhead).

Problem Definition

Given a set $v = \{v_1, v_2, .., v_k\}$ having k vehicles, it is required to choose the most efficient CHs to maximize the clustering efficiency (a composite, normalized metric that quantifies the effectiveness of a vehicular cluster in achieving key QoS objectives). This problem characterizes purely as an optimal decision problem. Since a CH can be selected in a number of ways, there should be a defined fitness function f (a measure of adequacy) because selecting any vehicle as the CH has a variable influence on clustering efficiency. The optimal vehicle can be identified as a CH by determining the value of a fitness function $f(v_i)$ for a vehicle v_i. Thus, the problem becomes selecting the vehicle that maximizes the fitness function. Additionally, in

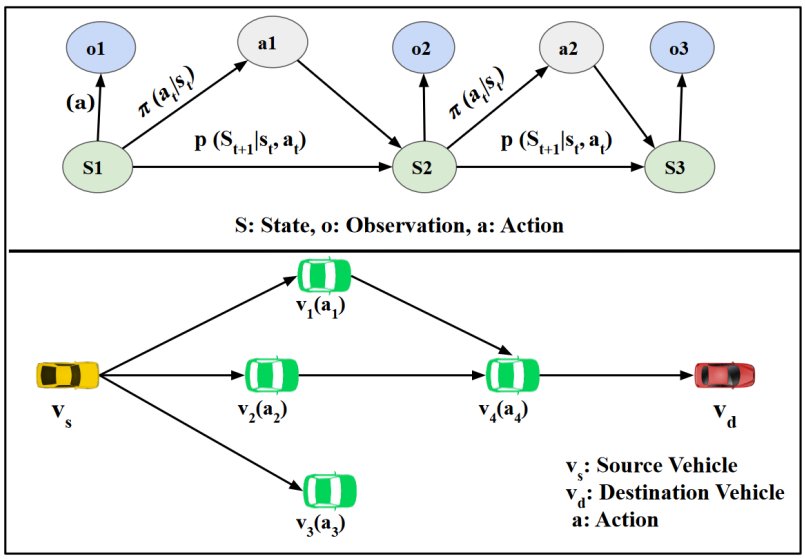


Figure 4.1: Routing problem with state-action pair

Q-learning, the Q function value identifies the route using transmission time to find the optimized routes. Q value for a source to destination state-action pair via intermediate vehicles is defined as $Q[(v_d, v_s), v_j]$, (throughout this chapter, the packet transmission convention will be used as [(destination, source), intermediate vehicle]) where the source vehicle v_s transmits the packet to the destination vehicle v_d using intermediate vehicle v_j. When a packet is transmitted to an intermediate vehicle through a link $v_s \rightarrow v_j$, the transmission time is observed, and an immediate reward is added for the given path. Figure 4.1, represent a vehicular routing problem, where v_s is the source vehicle and v_d is the destination vehicle. A packet transmitted to a destination using an intermediate link is considered as the action $A(v_d, v_s) = \{a_1, a_2, a_3\}$. In low mobility situations, Bellman's optimality principle [121] is employed to provide solutions by updating the Q value. However, in highly dynamic scenarios, it is necessary to modify the mechanism to ensure precise solutions.

Grey Wolf Optimization

GWO is one of several bio-inspired techniques that mimic the natural hunting behavior of grey wolves [122]. The grey wolf pack consists of 4 types of wolf hierarchy, namely *alpha*, *beta*, *delta*, and *omega*. Alpha wolves serve as the leaders of the hierarchy and are responsible for maintaining order within the pack. The beta wolves follow and support the alpha wolf in decision-making and assist in delivering commands to the lower-level grey wolf hierarchy. Delta is the third rank of the grey wolf. Their duties range from espionage to guarding, from hunting to caring for their prey. Omega grey wolves rank the last in the group and do not contribute much. A traditional GWO algorithm shows good optimization potential. However, it suffers from restricted exploration and a slow convergence rate [123]. The establishment and evolution of the grey wolf population may result in local optima or premature convergence for grey wolf locations and velocity updates. Therefore, a more effective optimization technique is needed for grey wolves to address the aforementioned challenges.

4.2.2 Clustering Process in MetaLearn

In the first phase, random clusters are created based on the number of vehicles and transmission range. Once initialized, vehicles are classified by features such as speed, direction, and position leveraging features from CSBR. To optimize clustering, a vehicle can belong to only one cluster at a time, with a CH coordinating operations. CHs also manage the addition and removal of vehicles.

Clusters Initialization

Initially, at the beginning of a journey, at time T, each vehicle reports its current state to nearby RSU and indicates whether or not it is a cluster member. The OBU evaluates network data, such as vehicle state data, to detect the presence of a CH. This query message will be transmitted to all relevant RSUs and vehicles. RSUs can identify instances of network activity by wirelessly connecting to other OBU devices in vehicles. When a CH is detected nearby, the vehicle will transmit a message to initiate the cluster merging procedure. A vehicle that is not currently part of a cluster must notify its immediate neighbors, who will then forward the request to the RSUs.

Cluster Formation

The cluster formation phase begins when a vehicle initiates a request message to other vehicles in the absence of a nearby CH. Upon receiving a request, vehicles are randomly assigned to a state space. The distance between vehicles and QoS parameters (e.g., link availability, queue occupancy, delay) are calculated to determine the fitness function. OBUs act as local controllers, evaluating the fitness function and determining cluster affiliation. Local computing minimizes route requests, with RSUs consulted only when no nearby vehicles are present. Vehicles interact with neighbors using beacon messages containing information on direction, position, and velocity. The clustering process of the grey wolf is detailed as follows:

Encircling: Grey wolves are capable of tracking and encircling their prey (optimum location). The alpha guides the search, with occasional participation from the beta and delta. However, in VANETs, the location of the optimal response are unknown in an abstract search space. So, the hunting behavior of grey wolves is recreated by assuming that the alpha (best candidate solution), beta, and delta have more knowledge of potential prey locations. GWO's procedure begins with the hunting pack's initialization. Let Lw_i, $1 \leq i \leq n$ initialize the locations of wolves (vehicles) involved in the hunt, where n specifies the number of agents/vehicles. A position vector $\vec{Lw}(t)$ is used to initialize the location of each wolf in t^{th} iteration. Wolf packs' social behavior is employed to investigate and encircle prey in the following manner:

$$\vec{Lw}(\vec{t}+1) = \vec{Lp}(t) - \vec{A} \cdot \vec{D}, \tag{4.1}$$

$$\vec{A} = (2\vec{a}.\vec{r_1} - \vec{a}), \tag{4.2}$$

$$\vec{D} = |\vec{CT} \cdot \vec{Lp}(t) - \vec{Lw}(t)|, \tag{4.3}$$

$$\vec{CT} = (2.\vec{r_2}), \tag{4.4}$$

Here, $\vec{Lp}(t)$ represents the current position vector of prey (optimum position for CH) in iteration t and $\vec{Lw}(t)$ represents the current location vector of wolf in iteration t. $\vec{A}$ & $\vec{CT}$ are the coefficient vectors defined in (4.2), (4.3), and (4.4) defines the distance between the prey and the wolf. The coefficient $\vec{A}$ declines linearly from 2 to 0 as the number of iterations increases, and the random vectors $\vec{r_1}$ and $\vec{r_2}$ are obtained within the

range [0, 1]. The linearization constant $\vec{a}$ is determined using (4.5), where N_{ctr} denotes the current iteration value and N is the total number of iterations.

$$\vec{a} = 2 - \frac{2N_{ctr}}{N}. \tag{4.5}$$

The key principles behind the position update methods in (4.1) and (4.3) are depicted in Figure 4.2.

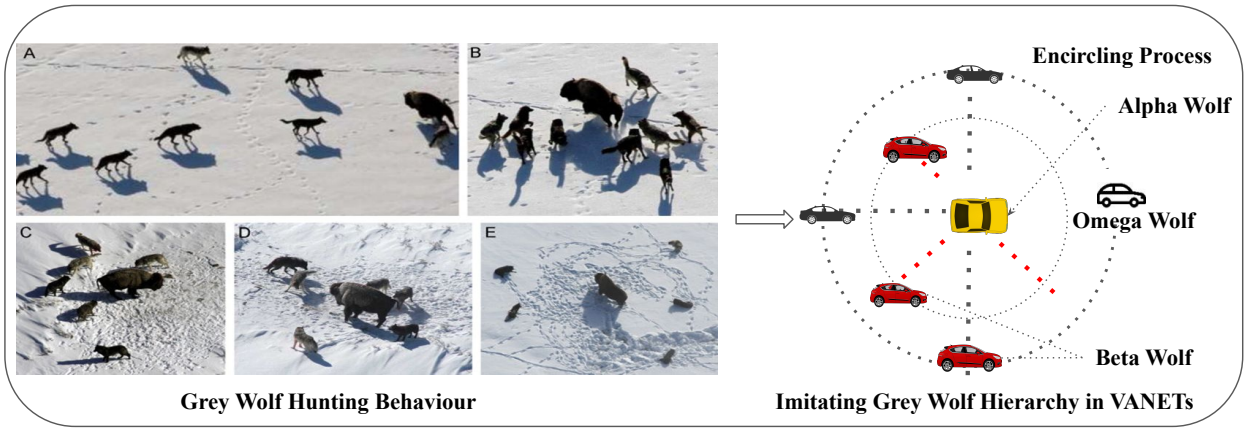


Figure 4.2: Grey Wolf Hierarchy (left) [124] and Encircling Strategy Adapted for VANET Clustering (right)

Exploration: In addition to the encirclement process, GWO needs extra operators to prioritize exploration. The alpha, beta, and delta locations serve as promising CH locations. The algorithm further seeks around to find optimum locations. According to Muro et al. [124], Figure 4.2 depicts the gray wolf's two-dimensional location in respect to its prey[1]. Suppose the wolf is located at (X, Y) and its prey is located at (X', Y'). Eq. 4.2 and 4.4 demonstrate that the grey wolf will follow its prey as the prey changes its location. Vectors $\vec{A}$ and $\vec{CT}$ are utilized to update the wolf position. The same position update is followed for a three-dimensional model where the wolf is at a random point in space (X, Y, Z) and the prey at another point (X', Y', Z'). The alpha, beta, and delta individuals are recognized as global best, second best, and third-best. On the other hand, omega wolves move in response to the presence of alpha, beta, and delta wolves in new areas. A set of solutions consisting of α, β, and δ is calculated at every step, and the location is updated as:

$$\vec{Lw}(t+1) = \frac{\vec{lw}_1 + \vec{lw}_2 + \vec{lw}_3}{3}. \tag{4.6}$$

The updated values of wolves is calculates as $\{\vec{lw}_1 = \vec{Lw}_\alpha - A_1.\vec{D}_\alpha\}$, $\{\vec{lw}_2 = \vec{Lw}_\beta - A_2.\vec{D}_\beta\}$, $\{\vec{lw}_3 = \vec{Lw}_\delta - A_3.\vec{D}_\delta\}$ and the distances is calculated as $\{\vec{D_{\alpha,\beta,\delta}} = |\vec{CT} \cdot \vec{Lp}(t) - \vec{Lw}(t)|\}$. The new possible solutions are searched based on the component values of α, β, and δ are presented in (4.7). Here, VS^{LB} and VS^{UB} are the lower and upper bounds for search state space.

$$Lw_i^j = \begin{cases} VS^{LB}, & \text{if } \{\alpha+\beta+\delta\}/3 \leq VS^{LB} \\ VS^{UB}, & \text{if } \{\alpha+\beta+\delta\}/3 > VS^{UB} \\ \{\alpha+\beta+\delta\}/3, & \text{otherwise..} \end{cases} \tag{4.7}$$

[1]Photograph courtesy: Erich Klinghammer, and Douglas Smith, Project Director of the Yellowstone Wolf Reintroduction Program, Yellowstone National Park, Wyoming U.S.A.

CH Selection: Following initialization, the OBUs calculate and store the fitness value for CH selection. The fitness function defines the network's QoS criteria and is computed using the link latency, distance, link availability time, and queue occupancy. Eq. (4.8) defines the fitness function.

$$Max[\omega_1 \cdot (1-LD) + \omega_2 \cdot (1-ED) + \omega_3 \cdot (LA) + \omega_4 \cdot (1-QP)], \tag{4.8}$$

Here, LD denotes link delay, ED denotes euclidean distance, LA denotes link availability, and QP denotes queue occupancy. The parameter ω_i denotes the weight constant in which $\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1$. The following equations can be used to determine the link status parameters that are utilized to offer QoS support in the network. The link availability function, computed using (4.9), is used to determine the time a connection is stable for communication.

$$LA = \frac{R_x - ED}{|vel_i - vel_j| + \delta}, \tag{4.9}$$

Here, R_x denotes the communication range and vel_i denotes the v_i vehicle's velocity. δ represent a minimum denominator threshold. ED denotes the euclidean distance, which is calculated using (4.10)

$$ED(\vec{v_i}, \vec{v_j}) = \sqrt{\sum_{r=1}^{N} (\vec{v_{ir}} - \vec{v_{jr}})^2}. \tag{4.10}$$

Link delay is calculated using (4.11), where T_s is packet sending time, and T_r is packet receiving time.

$$LD = \sum T_r - \sum T_s. \tag{4.11}$$

The OBUs calculate the queue occupancy value in order to determine the occupancy of packets in the queue to be processed.

Exploitation: The exploration and exploitation capabilities substantially influence GWO's search performance. This refers to wolves deviating from their original search path to hunt in a new direction, displaying their capacity to explore undiscovered places. Exploitation is the act of a wolf resuming its hunt for prey at a defined depth, ensuring that the wolf has thoroughly searched the area it has previously surveyed. To overcome this, it is necessary to balance exploration and exploitation by utilizing the two adaptive and random vectors $\vec{A}$ and $\vec{CT}$. When $\vec{A} > 1$ and $\vec{CT} > 1$, the wolf exhibits exploratory ability. On the other hand, $\vec{A} < 1$ and $\vec{CT} < 1$ increase the wolf's potential to exploit. The wolf's capacity to exploit its prey diminishes as the number of repetitions increases. Due to the random generation of the value of $\vec{CT}$, the wolf's exploration and exploitation may reach equilibrium at any point during optimization. It can avoid the algorithm becoming trapped in a local extreme during the iteration's final phases.

4.2.3 Route discovery using TD Learning

In the second phase, MetaLearn employs TDL to avoid local optima issues between the CH and the destination. The next-hop node for a data packet is determined by the sum of immediate rewards and multi-step error calculations. Hierarchical $Q(\lambda)$ learning reduces computation time and search space by allowing higher-level tasks to call lower-level tasks. The multilevel network structure divides route identification tasks, with source cluster tasks determining subtasks in intermediate clusters. In this clustering-based environment, packets traverse various subnetworks (clusters) to reach their destinations. The division of the state space into distinct clusters is as follows: (i) A packet from the originating vehicle v_s arrives at the alpha

node (CH) c_1^α in cluster c_1; (ii) The packet is transmitted from c_1^α to c_2^α in cluster c_2; (iii) The packet is then transmitted from intermediate node c_2^α to the destination node v_d in cluster c_3.

In the $Q(\lambda)$ learning model, the CH vehicle acts as the agent, with continuous states representing packet positions and destinations in the clustered network. Upon receiving a packet, each agent selects an action based on links towards the destination. MetaLearn uses multi-step TD(λ) learning to adapt the Q function more efficiently. The Q function value $Q[(v_d, v_s), v_j]$ estimates the transmission time for a packet from source v_s to destination v_d via intermediate vehicle v_j. When a packet is transmitted, an immediate reward $R[(v_d, v_s), v_j]$, a function of link delay time $T(v_s, v_d)$, is awarded. MetaLearn uses TD error to adapt the difference between the optimal and current Q values. To overcome single-step error observation in Q learning, MetaLearn employs aggregated multi-step error calculations for time instance $t+n(n = 1,2,3,...,T)$, providing faster adaptation for Q values. Initially, one-step TD errors are calculated using (4.12).

$$\phi_t = R[(v_d, v_s), v_j] + \gamma \underset{j \in A(v_d, v_s)}{Min} [Q[(v_j, v_s), v_k] - Q[(v_d, v_s), v_j]], \tag{4.12}$$

Here, $R[(v_d, v_s), v_j]$ is the immediate reward and the updated Q value function is calculated based on the $Min_{k \in A(v_j, v_s)} Q[(v_j, v_s), v_k]$ and the sum of immediate rewards. The updated Q value provides the approximation for future rewards. A parameterized approach removes future reward uncertainty by aggregating the value of the future reward to a discount factor γ. The eligibility trace approach, which is based on the time step updated methodology, is used to address the $TD(\lambda)$ error. To estimate the multi-step $TD(\lambda)$ error, many future actions are utilised. The $TD(\lambda)$ error ψ_t^λ is defined by (4.13) as the weighted sum of ψ_{t+n} for all $n = \{1, ..., T\}$. It is important to remember that the future error ψ_{t+n} is discounted by a decaying factor $\lambda\gamma$. A future TD error is more likely to be discounted when it is used to change the current Q function value. Future rewards can be included into the current computation of the Q function value, resulting in faster and more accurate estimates.

$$\psi_t^\lambda = \sum_{n=0}^{T} (\gamma\lambda)^n \psi_{t+n}. \tag{4.13}$$

The technique for performing multi-step $Q(\lambda)$ learning is provided by the eligibility trace notion. As additional reward R_{t+n} is realised, it helps to partially assess the value of ψ_t^λ. As a result, the current estimate of the current state-action pair ψ_t must be dispersed among the ψ_{t-n}^λ pairings. Furthermore, because recent state-action pairs have a greater impact on the present error ψ_t, it must be given more credit than from the past. It is thus necessary to apply the right ψ_t value to all Q function values to accurately record the state-action pairings. The eligibility trace $E_t(s,a)$ fine-tune the target values to improve the learning process. This means that the framework must reassess state-value function calculations for each state on a regular basis. Eligibility values close to or near zero for states that have not been visited in a significant period will have negligible influence on the estimates for those states. Decisions made by the agent at $E_{t-1}[(s,a)]$ will decay by a factor of $\gamma\lambda$ at each successive time step (t), and for every future action taken that changes the state, an $E_t[s,a]$ value will be added. Eq. (4.14) represents the eligibility trace at time t=0, and (4.15) represents the eligibility trace when an action is taken at time t.

$$E_0[(v_d, v_s), v_j] = 0 \ \ \forall v_j. \tag{4.14}$$

$$E_t[(v_d, v_s), v_j] = \gamma\lambda E_{(t-1)}[(v_d, v_s), v_j] + I_{v_d, v_s} \ \ \forall (v_d, v_s, v_j), \tag{4.15}$$

Here, I is the identity indicator function containing value of 1 if $v_d = v_s$ and 0 otherwise. Finally, after t+n time the Q value is updated as (4.16). The final Q value for all state-action pairs and every $Q[(v_d, v_s), v_j]$ is

added towards the error ψ_t and multiply by the learning rate α and the eligibility trace $E_t[(v_d, v_s), v_j]$.

$$Q[(v_d, v_s), v_j] = Q[(v_d, v_s), v_j] + \psi_t \alpha E_t[(v_d, v_s), v_j]. \tag{4.16}$$

Figure 4.3 illustrates a routing scenario where a multi-step Q value update is performed using backward propagation of value changes observed during routing decisions. The figure represents value propagation in a vehicular scenario for time steps $t = 1, 2, 3$. Initially, all states from source to destination are unvisited and possess an eligibility trace value of zero. At time $t = 1$, when a packet travels along the edge between vehicle v_s and vehicle v_1, the temporal difference (TD) error ψ_1 is calculated using $R[(v_d, v_s), v_1]$ and $Q[(v_d, v_1), v_2]$ based on equation (4.12). The eligibility trace value for the edge between v_s and v_1 is updated to $E_1[(v_d, v_s), v_1] = 1$, while all other traces remain zero. The TD error used to update the Q value is multiplied by the eligibility trace value. Consequently, the error value for $Q[(v_d, v_s), v_1]$ is ψ_1, whereas the values for other Q functions, such as $Q[(v_d, v_1), v_2]$ and $Q[(v_d, v_1), v_3]$, are zero. At time $t = 2$, as packets continue to traverse the network, the decay factor $\lambda\gamma$ comes into effect, and the eligibility trace $E_2[(v_d, v_1), v_3]$ increases by 1. The error applied to update Q values is $\lambda\gamma\psi_2$ for $Q[(v_d, v_s), v_1]$, ψ_2 for $Q[(v_d, v_1), v_3]$, and $\psi_2 = 0$ for $Q[(v_d, v_1), v_2]$. This procedure is applied to all state-action pairs available in the network. The

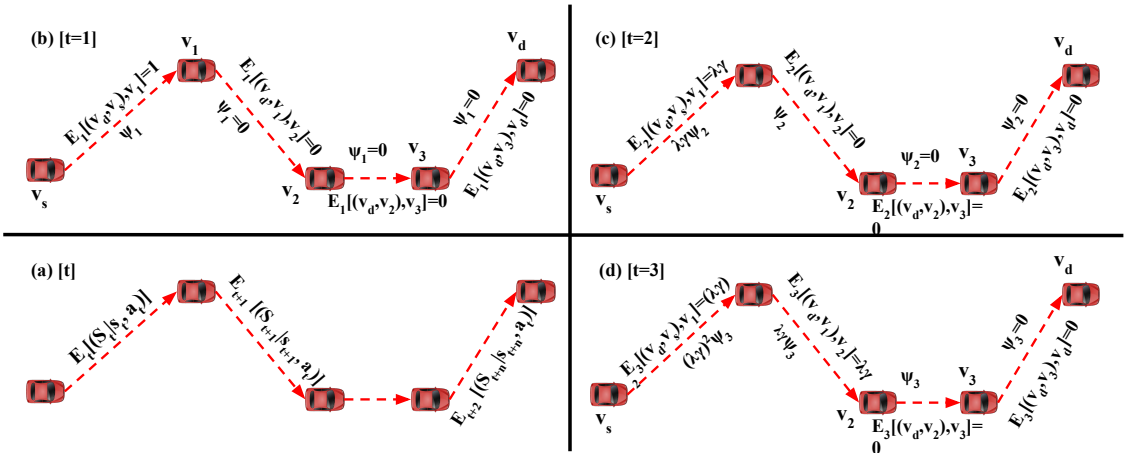


Figure 4.3: Eligibility trace and error propagation in MetaLearn

popular ε-greedy action selection technique in Q-learning is widely used in state-of-the-art methods to balance exploration and exploitation. However, it has a notable drawback: during exploration, it selects actions with equal probability, regardless of their projected value. Consequently, the worst possible action could be chosen with the same likelihood as the best one. This approach is inadequate for tasks where selecting the worst actions leads to significantly negative outcomes. Instead, action probabilities should be calculated based on their projected value. While all actions are ranked and weighted according to their assessed value, the greedy action is given the highest selection probability. These methods are referred to as "softmax" rules. MetaLearn employs the Boltzmann Distribution for action selection, choosing an action on the n-th iteration with probability:

$$P(v_d, v_s) = \frac{e^{-\frac{1}{\tau}\frac{Q[v_d, v_s], v_j}{Q[v_d, v_s]}}}{\sum_{v_k \in A} e^{-\frac{1}{\tau}\frac{Q[v_d, v_s], v_k}{Q[v_d, v_s]}}}, \tag{4.17}$$

Here, temperature (τ) is a positive variable. If the temperature parameter has a high value, the probability of selecting all neighbor nodes as the next forwarding relay will be equal. When the temperature has a low value, the likelihood of selection differs more for actions with various value evaluations.

Reward Function Setting

In MetaLearn, the QoS support is provided via information wrapped within the packet as proposed in OBQR [125]. Before forwarding a packet, each vehicle checks the QoS requirement. The vehicle updates the QoS criteria after each forwarding packet. The link status table at each vehicle is created using a weighted function to map the QoS requirements. This research considers a network's connection availability, path cost, and delay to give QoS support. Eq. (4.18) gives the weighted function.

$$w_i = \alpha \frac{LA_t}{LA_{tmax}} + \beta \frac{Ct_x}{Ct_{max}} + \gamma \frac{LD_x}{LD_{max}}, \tag{4.18}$$

Here, the parameter LA_t represents the link availability time, Ct_x is the cost associated with the path, and LD_x is the delay parameter. The parameters α, β, γ are the weight constants. The sum of weight constants $\alpha + \beta + \gamma = 1$. The unbounded network parameters are mapped into bounded parameters by converting the NP-complete problem to a polynomial time problem. The reward function is given by:

$$\mathbb{R} = \begin{cases} ED_{v_s v_d} - ED_{v_{im} v_d} / ED_{v_s v_d} \cdot min \sum_{i=1}^{k} w_i f_i(x), & \text{if } ACK = TRUE \\ -ED_{v_s v_{im}} / ED_{v_s v_d}, & \text{otherwise,} \end{cases} \tag{4.19}$$

Here, $ED_{v_s v_d}$ expresses the distance between the source and destination vehicle, whereas $ED_{v_{im} v_d}$ represents the gap between the intermediate vehicle and the destination vehicle. For ease of understanding, *Algorithm 6* represents the working of the MetaLearn routing optimization task.

Algorithm 6 MetaLearn routing optimization

Require: Q-value function $Q(s,a)$, state space S, action space A, reward r, discount γ
Ensure: Optimized routing paths

Input: $Q(s,a)$, S, A, r, γ
Output: Optimized routing paths
Initialize Grey wolf population $(Lw_i, 1 \leq i \leq n)$
while $Itr < Itr_{max}$ **do**
    for each $v \in VS$ **do**
        Compute $\alpha, \beta, \delta, \Omega$ values using (4.6)
    end for
    Update coefficients with (4.5), compute fitness with (4.8)
    Increment iteration: $Itr \leftarrow Itr + 1$
end while
if Beacon message timer expires **then**
    Broadcast beacon to neighbors, reset timer
end if
for each received packet **do**
    Set source v_s, destination v_d, initialize $t = 0$
    Choose relay node v_j via softmax policy, set $E_t[(v_d, v_s), v_j] = 0$
    while $v_s \neq v_d$ **do**
        Transmit packet to v_j, update eligibility and Q-value using (4.15), and (4.16)
        $t \leftarrow t + 1$
    end while
end for

Routing Implementation

The design and implementation of the MetaLearn protocol utilize the OpenAI Gym extension and NS3, enabling the execution of a RL environment within NS3. The NS3-Gym interface manages the simulation lifecycle, facilitating state and action data communication between the Gym agent and the simulation environment. Understanding MetaLearn's routing procedure requires comprehension of its use of Q-values. Beacon packet dissemination helps maintain local neighborhoods. Unlike reactive protocols that do not pre-identify routes, routes are stored in the routing table only after a node initiates route identification via a route request and receives a route reply. Upon receiving a route error message, the path can be restored by returning to the source node. If the sequence number is equal to or greater than that in the route request message, the newly discovered route is accepted as valid. This method communicates information on neighboring vehicles, including speed, position, and identity, to nearby nodes. Historically, beacons have been used for node discovery, but in MetaLearn, they also carry local Q-values of broadcasting nodes, indicating the likelihood of correctly relaying data across the network. The QVTab class maintains a node's transitive Q-value table, which is updated with Q-values from outgoing beacons. This table maps the node to others for which it has Q-value information. Additionally, geographical coordinates are included in the MetaLearn beacon header. Any received Q-values are immediately integrated into the node's transitive Q-value table. Beacons are modified to detect initial nodes, and after updating its Q-value table, a node sends an acknowledgment message.

4.3 MRMH: MULTI CONSTRAINT ROUTING OPTIMIZATION USING HYBRID META HEURISTICS IN VANETS

While MetaLearn significantly improves network performance through its hybrid of clustering and learning, certain limitations became apparent. MetaLearn's GWO-based clustering uses a linear parameter decay and static update rules, which may not adapt optimally under all conditions. For instance, in extremely dynamic scenarios, a faster or non-linear transition from exploration to exploitation might be needed for the heuristic to keep up. MRMH is an evolved framework that addresses these issues by introducing a more adaptive and multi-objective optimization approach. It builds on MetaLearn's foundation but incorporates two key innovations: (1) a hybrid optimization strategy combining GWO (global search) with SQP (local precise optimization), and (2) an expanded fitness function that explicitly accounts for multiple network performance constraints simultaneously (hence "multi-constraint").

4.3.1 Strategic Outline

The motivation for MRMH comes from recognizing that balancing exploration and exploitation is crucial for clustering and routing in VANETs, and that a single meta-heuristic algorithm might not achieve that balance perfectly on its own. MetaLearn's GWO, for example, uses a linear decrease of coefficient a from 2 to 0 over iterations. This works well in many cases, but if the optimal clustering requires quick adjustments (say, vehicles suddenly take a new road, requiring a new cluster formation), a linear schedule might be too slow to respond or could prematurely converge to a suboptimal clustering. Likewise, if multiple criteria (like minimizing delay and maximizing link stability and load balancing traffic) are to be met, GWO alone would require a very carefully crafted single fitness function or a multi-objective extension, which can complicate convergence.

MRMH introduces a nonlinear decay scheme for the GWO component's parameters and a weighted distance approach in the position update equations. In simpler terms, instead of reducing the randomness (exploration factor) at a fixed linear rate, MRMH adjusts it with a nonlinear formula that can make the algorithm stay exploratory longer or shorten the exploration phase depending on the scenario. For example, MRMH uses an exponential or adaptive decay for the a coefficient in GWO: if the network is changing rapidly (detected by large fitness fluctuations between iterations), a might not decrease too quickly, preserving the ability to explore new cluster configurations. If the network is relatively stable, a can drop faster to fine-tune the existing solution. This adaptive parameter control makes the heuristic more responsive to VANET volatility, essentially allowing it to, turn on a dime, when needed to chase a moving target (the optimal solution under changing conditions).

The second major aspect is hybridizing GWO with SQP. SQP is a deterministic mathematical optimization technique that excels at solving constrained optimization problems by iteratively approximating them as quadratic sub-problems. In MRMH, after a few iterations of GWO (or even at each iteration after candidate positions are updated), we invoke an SQP routine to refine the best solutions found. Essentially, the idea is: GWO provides details on which vehicles should be CHs and roughly how to route, then SQP takes that point and treats it as a starting point for solving a more exact optimization problem that includes the constraints and objectives we care about. Figure 4.4 illustrates the various phases[2] of the MRMH solution approach.

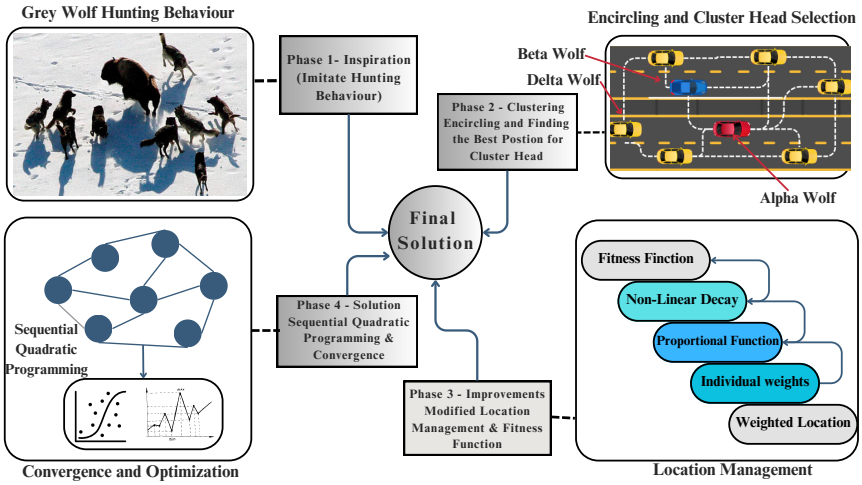


Figure 4.4: Flowchart illustrating the step-by-step process of MRMH's hybrid metaheuristic approach [124]

4.3.2 Problem Definition

Consider a VANET modeled as a graph $G = (V, E$ at a given moment. The set V represents vehicles $V = \{v_1, v_2, \ldots, v_n\}$, and E denotes bidirectional links connecting neighboring nodes $\{v_i, v_j\}$. These links, $E = \{(e_1, e_2), \ldots, (e_{n-1}, e_n)\}$, have a weight factor w indicating the quality of each path. The goal is to determine an optimal route from a source v_s to a destination v_d within this network, where $v_s, v_d \in V$, satisfying all QoS criteria. This constitutes a multi-objective optimization problem. We define feasible solutions, or viable

[2]The photograph copyright is subjected to Erich Klinghammer, and to Douglas Smith, Project Director of the Yellowstone Wolf Reintroduction Program, Yellowstone National Park, Wyoming U.S.A.

paths ρ, as elements of the set of all paths P in G. Each solution is expressed as $Y \to f(x) \in X \subseteq Sol^z$, where the functions yield positive rational values and are computationally feasible. For each objective i within a range of 1 to z, rational bounds are set as $LB(z)$ and $UB(z)$:

$$LB(z) \leq f(x) \leq UB(z) \ \ \forall x \in X. \tag{4.20}$$

In addressing the multi-constraint optimization challenge with weight vectors $w_i = (w_1, w_2, \ldots, w_z) \in Sol^z > 0$, we apply the weighted sum scalarization:

$$\max \sum_{i=1}^{z} w_i \cdot f_i(x), \tag{4.21}$$

Here, (4.21) represents a polynomial-time approximation ψ that simplifies the NP-complete problem into a manageable format, where $\psi \geq 1$ relates to the size of the input.

Mobility Model

MRMH simulates the urban vehicular communication with a more realistic mobility model than MetaLearn. The vehicle set $V = \{v_1, v_2, \ldots, v_N\}$ includes vehicles equipped with OBUs characterized by processing capabilities P_{obu} and memory capacity M_{obu}. RSUs, represented by $R = \{r_1, r_2, \ldots, r_M\}$, are strategically positioned with advanced processing P_{rsu} and storage capacities M_{rsu}. Vehicular communication utilizes C-V2X technology over the sub-6 GHz spectrum with a bandwidth B_{v2x} for both V2V and V2I interactions. While C-V2X allows direct vehicle communication, its range is limited. RSUs, with higher power and advanced antennas, extend this range. The *PC5* interface enables direct device communication, bypassing base stations, which is crucial for latency-sensitive applications and rapid data exchanges vital for vehicular safety. Additionally, the User Equipment to *Uu* interface supports communication with cellular networks. In this bidirectional communication model, OBUs transmit their status and receive directives, while RSUs process this data and respond accordingly. Both OBUs and RSUs operate within a predefined communication radius, ensuring consistent and reliable communication. The technical implementation of MRMH is depicted in Figure 4.5. The illustration outlines a structured approach to optimization, leveraging sequential updates, clustering metrics, and iterative improvement to find the best solution. The algorithm begins with generating an initial solution and updates the solution based on several factors, including parameters like alpha, beta, delta, and omega, which guide the update process. The solutions are iteratively moved towards the best-found solution, evaluated by a fitness function. The algorithm balances exploration and exploitation to refine the optimization process. Sequential programming is used for updating locations step-by-step. Metrics such as cluster density, inter-cluster distance, and orientation similarity are considered for evaluating clustering quality.

4.3.3 Proposed Methodology: MRMH

This section discusses MRMH's methodology, detailing each step of the process. At any specific time T, vehicles maintain their current state and cluster affiliations. OBUs scan the network for nearby CHs and communicate their findings to corresponding RSUs or vehicles. RSUs, in turn, maintain an updated network status by interfacing with various OBUs. Upon detecting a nearby CH, the protocol initiates a cluster integration process. If no proximate CH is found, the vehicle initiates cluster formation, inviting neighboring vehicles to connect and establishing links with relevant RSUs and vehicles.

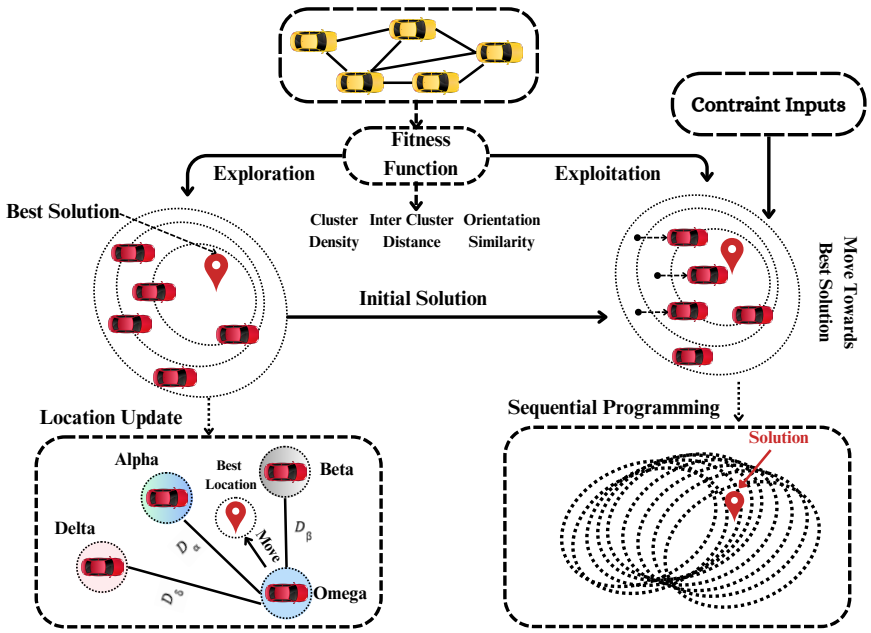


Figure 4.5: Technical implementation of MRMH

Location Modification Strategy

This enhancement is designed to expand GWO's exploratory capabilities. MRMH implement a sophisticated position-updating equation based on the weighted distance of each wolf, supplemented by an elimination and repositioning strategy. This approach broadens GWO's search potential. The algorithm's global exploration capacity is influenced by the leadership trio (alpha, beta, and delta wolves), with the vector coefficients $\vec{A}$ and $\vec{C}$ determining the alignment between omega wolves and the leadership wolves. In traditional GWO, the $\vec{A}$ coefficient, related to the hunting wolves' weights, can lead to redundancy issues. MRMH propose a novel weighted distance to address this:

$$W_i = \frac{W_1 L_\alpha + W_2 L_\beta + W_3 L_\gamma}{3}. \tag{4.22}$$

To better tackle complex, high-dimensional challenges, MRMH refine the position-updating equation with varying weights across iterations, integrating essential data:

$$W_1 = \frac{f_\alpha}{f_\alpha + f_\beta + f_\gamma} F(0), \tag{4.23}$$

$$W_2 = \frac{f_\beta}{f_\alpha + f_\beta + f_\gamma} F(1), \tag{4.24}$$

$$W_3 = \frac{f_\gamma}{f_\alpha + f_\beta + f_\gamma} F(2), \tag{4.25}$$

Here, $\{F(X)|X = 0,1,2\}$ signifies the proportionally weighted function used to leverage the fitness values of different wolves within the GWO hierarchy. The function is defined as:

$$F(X) = \frac{1}{e^{X-1}/\varepsilon}, \tag{4.26}$$

where ε is the trajectory scaling factor, crucial for wolves during position updates.

Nonlinear Decay for Enhanced Convergence

In metaheuristic algorithms, particularly in GWO, balancing exploration and exploitation is crucial for locating the global optimum [126]. The control parameters $\vec{A}$ and $\vec{C}$ in GWO are navigational parameters in this optimization, guiding the algorithm to either explore broadly (high values) or focus narrowly (low values) [127, 128, 129]. The linear decline of $\vec{A}$, a standard in traditional GWO, may not adequately encapsulate the algorithm's nuanced search behavior. Therefore, MRMH introduce a dynamic, nonlinear approach to manage these parameters. In VANET scenarios, characterized by complex and rapidly changing network dynamics, nonlinear decay mechanisms can significantly enhance route optimization. This approach allows the algorithm to adapt swiftly to changing network conditions, thus improving its efficiency in finding optimal routes. The introduction of a nonlinear decay factor ϕ, depicted as $\phi = \frac{MaxItr-t}{MaxItr}$, modulates the algorithm's progression, making it highly responsive to the evolving VANET environment. As the value of $\vec{A}$ oscillates within the interval $[-1,1]$, it dictates the algorithm's focus—below one targets the 'prey', while above one encourages exploration for potentially superior solutions.

$$\vec{a} = \phi a'(t) + (1-\phi)a"(t). \tag{4.27}$$

The decay factor ϕ governs the transition from exploration to exploitation as:

$$\vec{a'} = a_{in}(e^{-\lambda(a_{in}-a_{fin})}), \tag{4.28}$$

$$\vec{a"} = a_{fin}\left(\frac{MaxItr-t}{MaxItr}\right)^{M}, \tag{4.29}$$

where M, the ModIndex, is calculated as $\frac{f(\alpha)+f(\beta)+f(\gamma)}{3} - F(X)$, reflecting the average performance of the leading wolves, thereby guiding the algorithm's adaptive behavior in the complex VANET landscape. Further, the parameters a_{in} (initial value of a) and a_{fin} (final value of a) are parameters that define the starting and ending points of the decay process.

4.3.4 Clustering Process and Fitness Measure

In the clustering process within VANETs, identifying optimal locations for CHs is crucial. This is achieved using a fitness measure that incorporates several key factors:

$$g(x) = [(D_c) + S_c + (1-I_c)]. \tag{4.30}$$

This fitness function $g(x)$ combines cluster density (D_c), vehicle orientational similarity (S_c), and the inverse of inter-cluster distance ($1-I_c$), thereby providing a holistic measure of a potential CH's effectiveness. The inter-cluster distance I_c is critical for evaluating the suitability of a CH. It is calculated as the average

Euclidean distance between a vehicle and its neighbors:

$$I_c = \frac{\sum_{j=1}^{Nb_i} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{Nb_i}, \tag{4.31}$$

where Nb_i is the number of neighboring nodes for vehicles v_i. A lower I_c indicates closer proximity among vehicles in a cluster, which is desirable for effective communication. The orientational similarity S_c assesses how similarly vehicles within a cluster are moving, using CSBR's [130] cosine similarity approach. Finally, the cluster density D_c reflects the number of vehicles within a cluster. A higher density can lead to better coordination and data sharing among vehicles. Optimizing these measures - inter-cluster distance, vehicle orientation similarity, and cluster density - is pivotal in determining the most suitable CHs for efficient routing in VANETs, ensuring reliable and streamlined communication.

4.3.5 Hybridization of Heuristics

SQP plays a pivotal role in nonlinear optimization, particularly for VANET clustering. Known for its fast convergence and computational efficiency, SQP is adept at handling the exploitation phase in the MRMH. Its algorithmic robustness facilitates efficient navigation through complex VANET solution spaces, identifying optimal routing paths more rapidly. SQP's integration in the GWO cycle involves formulating and solving quadratic subproblems that consider both linear constraints and approximated Lagrangian functions:

$$\mathscr{L}(x, \lambda) = g(x) + \sum_{i=0}^{z} (I_i(x), \lambda_i), \tag{4.32}$$

Here, $g(x)$ is the fitness function, and $I(x)$ represents the inequality constraints, with z as the number of constraints. Post each GWO iteration, SQP refines the solution, initially creating interconnected clusters through CHs. The preliminary cluster assignments are based on randomized routes, which are then iteratively improved. MRMH first selects a cluster, performs a local search with SQP, and generates a new solution. If this new solution offers better efficiency, it replaces the previous one. This process is repeated until an optimal solution emerges or the maximum iteration threshold is reached. The objective of exploring the solution space is defined as:

$$f(h(sol)) = \max\{h(sol_x) | sol_x \in v, v \subseteq VS\}. \tag{4.33}$$

Upon finding a superior solution, the algorithm adapts according to the following criteria:

$$h(sol_{x'}) \geq h(sol_x), ; \{\forall sol_x \in v\}. \tag{4.34}$$

SQP iteratively solves the subproblem reflecting the local characteristics of nonlinear programming:

$$h(sol_x) + h(sol_{x'}) + \bigtriangledown h(sol_{x'})(sol_x - sol_{x'}) \ + \frac{1}{2}(sol_x - sol_{x'})^T M h(sol_{x'})(sol_x - sol_{x'}), \tag{4.35}$$

Here, $M(sol_x)$ represents the Lagrange multiplier, estimated using the Quasi-Newton update method. The subproblem is further simplified as:

$$\min \bigtriangledown h(sol_{x'})^T A(x) + \frac{1}{2} A(x)^T B(x) A(x). \tag{4.36}$$

This method continues to seek the optimal solution for $A(x)$ through a line search until convergence is achieved. The Hessian matrix of the Lagrangian function, derived from the GWO objective function, contains crucial second-order data for constraint functions, enhancing the search process's effectiveness. MRMH methodology in *Algorithm 7* commences with vehicular space initialization, followed by iterative position and fitness value calculations for the alpha, beta, delta, and omega wolves, representing network vehicles. Adhering to GWO principles, the algorithm updates coefficients for balanced exploration and

Algorithm 7 Proposed MRMH algorithm

Input: Locations $\vec{Lw}_i^j(t)$
Output: Optimal Locations $W^\alpha, W^\beta, W^\delta$
Initialize: Vehicular Space $v = \{v_1, v_2, ..., v_n\}$
while $Itr < Itr_{max}$ **do**
    for each $v \in v$ **do**
        Calculate positions for $\alpha, \beta, \delta, \Omega$ Wolves using (4.22)
        Calculate fitness value using (4.30)
    end for
    Update coefficients using (4.2), (4.4), (4.27)
    Generate initial solution $Sol_0 \leftarrow InSol()$
    for $Itr < Itr_{max}$ **do**
        Apply SQP to find optimal solution
        if $f(sol_{x'}) > f(sol_x)$ **then**
            $sol_x \leftarrow sol_{x'}$
        end if
    end for
    if Current Cost = Last Iteration Cost **then**
        Terminate and **return** $W^\alpha, W^\beta, W^\delta$
    end if
    $Itr \leftarrow Itr + 1$
end while
return $f(sol_x)$

exploitation. An initial solution is crafted using random neighbor structures, setting the stage for iterative optimization. SQP then refines the GWO solution, updating it if a superior one is identified. The process halts once the solution stabilizes or the iteration limit is reached. The algorithm culminates by pinpointing the optimal positions of the alpha, beta, and delta wolves, indicating the network's most strategic vehicle placements.

4.4 EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate the MetaLearn and MRMH frameworks, we developed a simulation framework that mirrors realistic vehicular network conditions. The simulation environment was designed with the following goals in mind: (a) to use realistic mobility patterns and traffic scenarios so that results reflect real-world behavior, (b) to implement standard VANET communication models (like C-V2X) to accurately capture wireless performance, and (c) to test a variety of conditions (vehicle densities, speeds, message loads) to validate the

robustness of our approaches. We also compare the proposed frameworks against representative baseline protocols from the literature to quantify improvements.

Experimental Setup

For the mobility and traffic simulation, we utilized the SUMO tool [43], which allowed us to generate real-world traffic patterns. We imported a city road map (Jodhpur, India, covering an area of 6000 x 6000 m^2 [113]) from OpenStreetMap to ensure that our scenarios include realistic road layouts, intersections, and traffic control (traffic lights, stop signs). SUMO was configured to produce typical urban traffic flows: during peak periods, vehicles experience stop-and-go traffic and congestion at intersections, while off-peak periods have free-flow traffic. We also simulated a highway scenario by including a highway segment connected to the city, where vehicles travel at higher speeds with fewer stops. This combination of city and highway mobility allowed us to expose the protocols to both low-speed dense conditions and high-speed sparse conditions. Vehicle movement patterns were derived from SUMO's car-following models [131] and route choices, providing realistic accelerations, decelerations, and lane-changing behavior.

The network simulation was carried out using NS3, which we integrated with SUMO via the Traffic Control Interface (TraCI) to have real-time mobility updates. We used the 802.11p WAVE [15] standard for the MAC and physical layer, with NS-3's YansWifiPhy and Wifi80211pHelper classes to model the wireless channel at 5.9 GHz. The propagation model included path loss and fading characteristics typical for urban environments (with parameters chosen to reflect an open space range of around 300 m. We also enabled NS-3's physical layer packet error models, meaning that interference and collisions could cause packet loss, not just distance. This is important to realistically assess reliability (PDR). Each vehicle's OBU was simulated as an NS3 node running our routing protocol on top of UDP/IPv4. Beacon messages were sent by each vehicle every 1 second (1 Hz beacon rate) to share their position and velocity with neighbors, which our protocols use for clustering decisions. Data traffic was generated in the form of periodic broadcast or unicast messages depending on the experiment.

We varied the number of vehicles in the simulation from around 50 (sparser network, akin to light traffic) up to 200 (very dense traffic). A medium scenario of 100 vehicles in the city map corresponds to a moderately congested urban area. This range allows us to see how the algorithms scale – e.g., whether many clusters vs. few clusters are formed and how overhead grows with more nodes. Both urban and highway speed profiles were considered. In the city portions, vehicles moved with typical urban speeds of 30–50 km/h (with frequent stops), whereas on the highway segment some vehicles reached 100–120 km/h. We also tested different messaging rates. In one set of experiments, each vehicle generated a safety message (e.g., a basic safety message with location info) every 0.5 seconds to be broadcasted, this is a high load scenario particularly in dense networks. By adjusting load, we examine how the protocols perform under network stress (do they maintain reliability when the channel is busy?). We included scenarios both with and without RSUs. In RSU scenarios, we placed a few RSUs (e.g., 4 RSUs) at strategic intersections in the city map. These RSUs are connected to a backbone (which we simulate simply as ideal links between RSUs for range extension). The protocols can use RSUs as super-nodes for clustering or relaying. This lets us evaluate if our frameworks can integrate V2I communication and whether it improves performance (it often does by providing an alternate path). In no-RSU scenarios, everything is purely ad-hoc (V2V). To validate the reliability of the results, multiple simulations were conducted with varying vehicle counts and different random seeds across 75 runs, accounting for inherent variability in initial conditions and traffic patterns, thereby enhancing the robustness of the findings. Within the GWO segment, 100 vehicles were

initialized, with the positions of key wolves—alpha, beta, and delta—dynamically updated based on fitness evaluations. Additionally, we calibrated key simulation variables (as outlined in Table 4.1) to ensure that the simulation conditions realistically reflected the operational constraints and variability typical of VANETs.

Table 4.1: Simulation Parameters for MRMH Experimental Analysis

Parameter	**Value/Range**	**Parameter**	**Value/Range**
Network Simulator	NS-3.38	**Operating System**	Ubuntu 20.04 LTS
Processor & Memory	Intel Core i7, 16 GB RAM	**Network Size**	6000m x 6000m
Vehicle Count	20–200	**Node Placement Model**	Random, Grid-based
Mobility Model	Random Waypoint (SUMO)	**Vehicle Speed Range**	20–100 km/h
Acceleration Model	SUMO Car-Following Model	**Communication Range**	250–400 meters
Transmission Power	0.1–1.0 W	**Packet Size**	256–1024 bytes
Traffic Load	5–30 packets/s	**Simulation Duration**	600–1800 seconds
Propagation Model	Two-Ray Ground, Nakagami-m	**Channel Model**	C-V2X
Path Loss Model	Log-Distance Path Loss	**Environmental Obstacles**	Enabled
MAC Protocol	IEEE 802.11p MAC	**Energy Consumption Model**	Node-level Tx/Rx
Inertia Weight (GWO)	0.6	**Decreasing Factor (GWO)**	[2,0]
Lower Bound (GWO)	0	**Upper Bound (GWO)**	100
Number of Lanes	4	**Total Simulations**	75
Statistical Analysis	Friedman and Holm's Post-Hoc	**Data Analysis Platform**	MATLAB 2021b

State-of-the-art Selection and Comparative Analysis

To evaluate the effectiveness of the proposed MRMH algorithm, we performed a comparative analysis against a suite of state-of-the-art metaheuristic algorithms widely applied in VANET optimization. Benchmark protocols included ACO and GWO-based methods, specifically ACO-AODV [77] and GSO-RBVT-R [78], which are recognized for their contributions to routing stability but exhibit latency due to their inherently sequential search mechanisms. Additionally, hybrid algorithms such as MetaLearn [132], which integrates GWO with Q-Lambda learning, and MOCRAW [80], which combines Dragonfly Algorithm and ACO for multi-objective clustering, were assessed. These hybrid models demonstrate improvements in path stability and network throughput but are constrained by substantial computational overhead and a predisposition to premature convergence. Further analysis included recently developed algorithms i.e., Rajput et al., [81], such as Firefly-Cuckoo Search (FA-CS) designed for adaptive path selection, Improved African Buffalo Optimization with Whale Moth Flame Optimization (MHCRT) [82] targeting energy-efficient clustering, and Giraffe Kicking Optimization (GKO) [79] with C-means clustering to enhance exploration-exploitation balance. While these algorithms address critical energy and adaptability challenges, they exhibit limitations in highly dynamic and high-velocity VANET contexts. In contrast, the proposed MRMH algorithm leverages GWO's global optimization capabilities with an innovative nonlinear decay function, complemented by SQP for rapid local optimization. This synergy facilitates a superior trade-off between delay, packet delivery ratio, and computational efficiency, thereby enabling MRMH to outperform existing methodologies in complex, real-time VANET scenarios.

Network Performance

In evaluating network performance, average transmission delay is a critical metric influenced by vehicle count and speed. Figures 4.6a and 4.6b illustrate this relationship, providing insights into the network's

effectiveness. The proposed MRMH algorithm significantly outperforms its predecessors by substantially

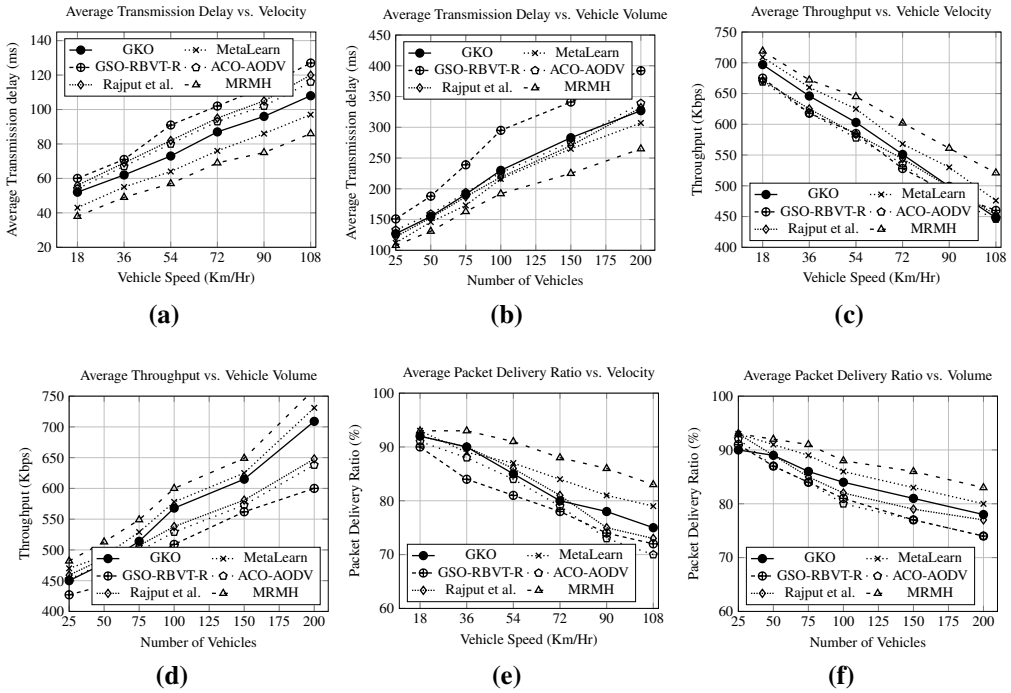


Figure 4.6: MRMH's experimental results of network performance

reducing average transmission delay. This improvement is attributed to MRMH's proficiency in handling diverse vehicular attributes during CH selection. By leveraging the hierarchical hunting dynamics of grey wolves, MRMH effectively identifies optimal CHs and relay nodes based on vehicular positions and trajectories, ultimately selecting a sustainable CH with the highest fitness value. This robust CH selection mechanism allows MRMH to decrease average transmission delay. In contrast, ACO-AODV underperforms due to its numerous competing agents and the slow convergence inherent in its sequential processing, compounded by iterative fitness evaluations, making it more time-intensive. Figures 4.6c and 4.6d illustrate the system's average throughput as influenced by vehicle count and speed. The MRMH algorithm excels due to its effective balance of exploration and exploitation during CH selection. This efficiency stems from strategic path selection, optimizing vehicle proximity durations, and enhancing bidirectional communication. Additionally, vehicle trajectories significantly impact cluster distribution, resulting in optimal local solutions that integrate into a robust global solution across all clusters. Figures 4.6e and 4.6f illustrate the PDR. MRMH's CH selection, based on a thorough evaluation of vehicle attributes such as location, direction, and speed, ensures optimal positioning for intra-cluster communication, thereby minimizing packet loss. By incorporating SQP during its exploitation phase, MRMH addresses quadratic approximations of the primary problem, optimizing route selection for packet transmission and enhancing the PDR. However, as the velocities increase, this ratio tends to decrease due to the higher likelihood of route changes and intersections, which can impede packet transmission.

Cluster Stability

Cluster stability is a crucial metric for network performance and a key evaluation criterion for MRMH clustering methodology. Achieving stability requires minimizing transitions for both CHs and members, necessitating a clustering mechanism that effectively regulates CH replacements to maintain cluster coherence. Figures 4.7a, 4.7b, and 4.7c illustrate CH performance across various speeds, vehicle densities, and transmission rates. As vehicle speeds increase, the average CH lifetime decreases, highlighting the difficulty of

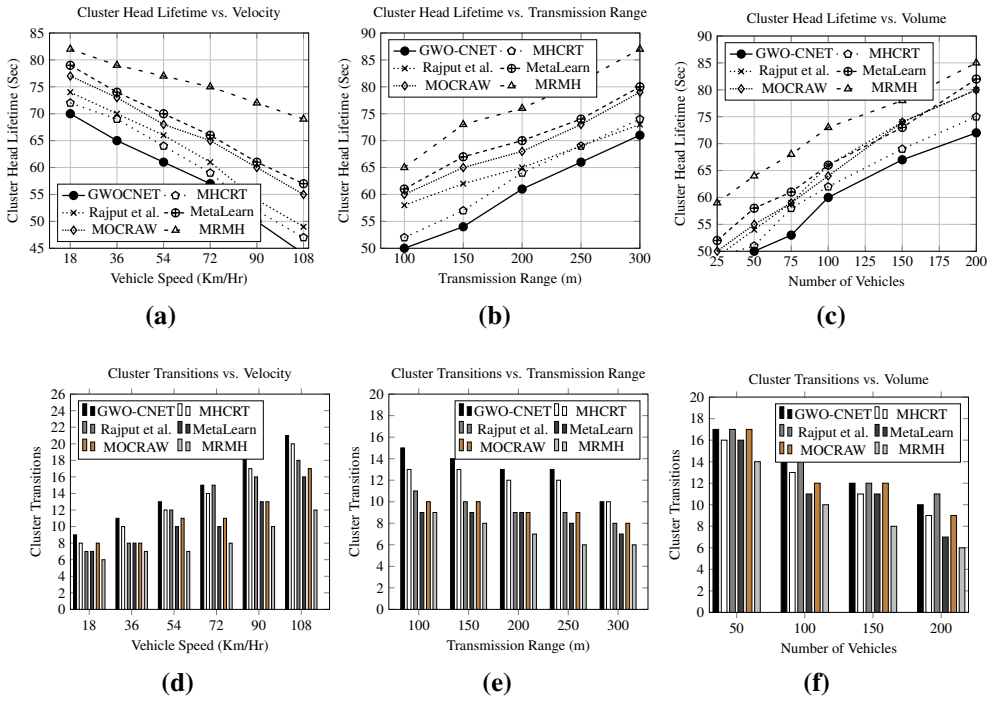


Figure 4.7: MRMH's experimental results for cluster head lifetime and cluster transitions

sustaining stable interactions with neighboring vehicles. While MRMH, MetaLearn, and Rajput et al. show extended lifetimes at lower speeds, their effectiveness diminishes at higher velocities. GWO-CNET consistently underperforms due to its inadequate handling of CH transitions and frequent topological changes. GWO based algorithms leverage location shifts observed during foraging. Initial placements rank wolf locations by fitness, and with the prey's location uncertain, three predictive scenarios guide wolf movements. Focusing on the top three global solutions, GWO enhances vehicle clustering and CH selection capabilities. MRMH's inter-fitness-based neighbor selection mechanism surpasses existing standards in identifying stable neighbors. Additionally, a broader transmission range correlates with longer CH lifetimes, as closer vehicles interact more effectively, fostering stable link establishment. Figures 4.7d, 4.7e, and 4.7f illustrate the cluster transition rate across various speeds, densities, and transmission ranges. As velocity increases, the transition rate rises due to rapid changes in network topology, resulting in member departures or mergers. In contrast, an extended transmission range enhances intra-cluster communication precision, thereby reducing transitions. MRMH's transition rate is comparable to MetaLearn's at both low and high transmission ranges; however, MetaLearn's rate increases significantly with higher cluster velocity. GWO-CNET's performance declines at higher velocities. The protocol by Rajput et al., which leverages swarm intelligence, aims for optimal solutions but suffers from prolonged convergence times, hindering effective global

maximum identification. Notably, larger clusters and improved connectivity at higher transmission ranges result in fewer transitions.

Test Suits for Efficiency Assessment of GWO-SQP Hybridization

The efficiency of the GWO-SQP hybridization is assessed using mathematical functions with well-defined global optima, as described in [122]. Optimization tests were conducted using Matlab R2021a with a standardized population size of 50 and 500 independent runs per experiment, ensuring consistent maximum objective function evaluations across all methods. Eight benchmark functions, detailed in Table 4.2, were utilized and categorized into unimodal, multimodal, and fixed-dimension functions. Unimodal functions,

Table 4.2: Benchmark functions definition

Function	Value
F1	$F_1(x) = \sum_{i=1}^{D} x_i^2$
F2	$F_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$
F3	$F_3(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
F4	$F_4(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
F5	$F_5(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$
F6	$F_6(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$
F7	$F_7(x) = -\sum_{i=1}^{7} \left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$
F8	$F_8(x) = -\sum_{i=1}^{10} \left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$

characterized by a single global optimum, measure the algorithm's exploitation capabilities. In contrast, multimodal functions, containing multiple local optima, assess the algorithm's exploration abilities and robustness against local optimization traps. Figures 4.8 illustrate the convergence curves of benchmark functions for competitive optimization, highlighting the superior performance of the hybrid GWO-SQP over standard GWO. The rapid convergence rate of SQP enhances the refinement of global optima, reducing the need for extensive random searches. Table 4.3 shows that GWO-SQP excels in unimodal benchmark functions, with results for F1 and F2 (Figures 4.8a and 4.8b) comparable to those of GWOCNET, underscoring its proficiency in exploitation. In multimodal test functions, GWO-SQP demonstrates superior exploration capabilities, effectively navigating numerous local optima. The algorithm's extensive probing of the search space and ability to avoid local traps are evident. GWO-SQP consistently produces results with lower standard deviation than GWO, indicating higher reproducibility. Notably, in fixed-dimensional multimodal functions F5 and F6 (Figures 4.8e and 4.8f), GWO-SQP outperforms GWO even when the latter shows a slightly higher standard deviation. For functions F7 and F8 (Figures 4.8g and 4.8h), GWO-SQP's results are closer to the global optimum, further affirming its better performance over state-of-the-art methods.

Fitness Function Analysis

Figure 4.9 presents the fitness analysis. The figure provide a visual comparison of the effectiveness of these techniques under different traffic densities. It highlights how each method performs in terms of the fitness function, which likely measures the efficiency or optimization quality of the routing paths. The algorithm shows a significant increase in fitness value with each iteration. When benchmarked against GWOC-NET,

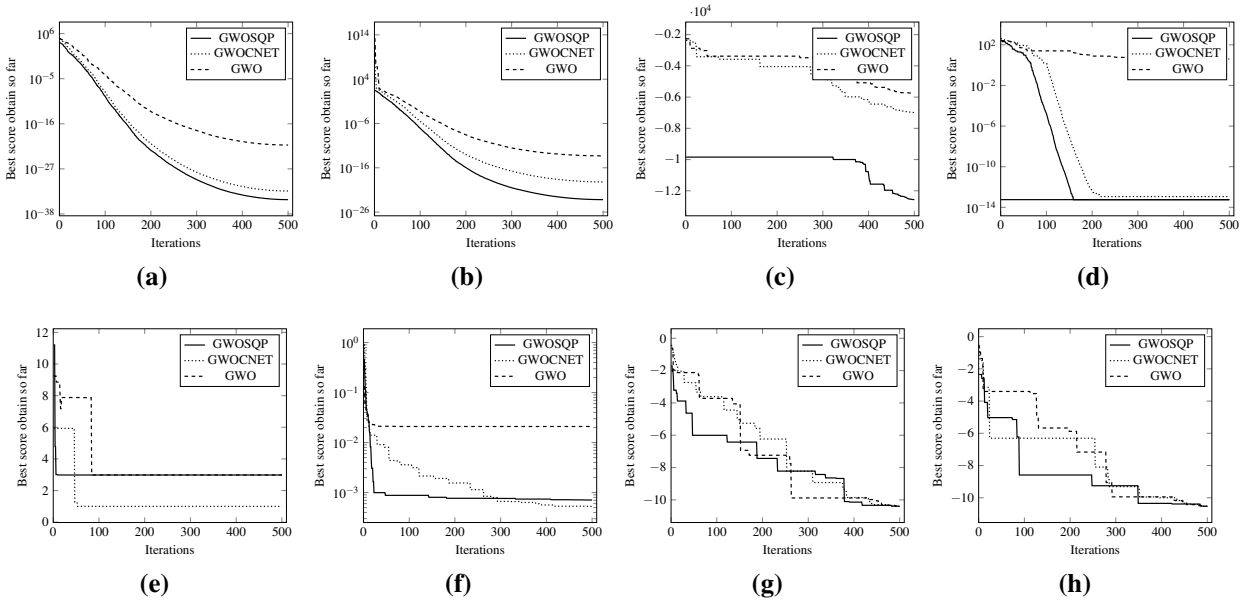


Figure 4.8: Convergence Graph of Benchmark Functions: (a) and (b) Unimodal Functions, (c) and (d) Multimodal Functions, (e), (f), (g), and (h) Fixed-Dimension Functions

Table 4.3: Convergence analysis of benchmark functions vs MRMH

Function	GWO-SQP			GWO-CNET			GWO		
	Best	Median	Worst	Best	Medium	Worst	Best	Median	Worst
F1	3.940E-37	3.073E-35	2.095E-33	2.048E-36	2.481E-33	2.583E-30	2.029E-17	2.002E-15	2.0381E-13
F2	4.028E-24	3.958E-20	4.009E-17	3.897E-20	3.984E-17	3.743E-16	2.539E-14	2.953E-11	2.532E-08
F3	-12566.319	-11984.4132	-10431.531	-6988.4123	-6732.987	-6612.192	-5756.31	-5498.1231	5192.1983
F4	5.31E-14	5.1E-13	5.43E-12	1.17E-13	1.18E-11	1.98E-09	4.92E+00	4.12E+00	3.87E+00
F5	2.8712	2.4561	2.1382	0.9983	0.7861	0.5642	2.8689	2.7698	2.5671
F6	7.908E-3	7.593E-3	7.120E-3	5.31E-3	5.10E-3	4.98E-3	20.43E-1	17.41E-1	14.12E-1
F7	-10.5392	-10.3231	-10.1431	-10.2321	-10.1012	-9.8974	-10.1043	-10.0421	-9.7898
F8	-10.5573	-10.4127	-10.3907	-10.5481	-10.3841	-10.1874	-10.5412	-10.4731	-10.3207

Rajput et al., and MetaLearn, MRMH consistently outperforms these methods due to its superior ability to evade local maxima and its robust exploration strategy. Rajput et al., which combines a multi-objective firefly algorithm with PSO optimization, demonstrates slower convergence compared to GWO. Over 100 iterations and vehicle densities ranging from 25 to 200, MRMH starts with a modest fitness value but consistently improves as iterations progress.

Statistical Analysis

Table 4.4 presents a statistical analysis from ten runs of various metaheuristic methods and MRMH for a VANET instance, focusing on a medium-density scenario (100 vehicles at 60 km/hr) to simulate urban and highway traffic. MRMH exhibits significantly lower time complexity, with a mean time of 1.376, a best-case scenario of 1.002, and a median of 1.673, outperforming other techniques. Its low standard deviation (0.172) indicates consistent efficiency, which is crucial in VANET environments where rapid response times are critical. In comparison, ACO-AODV shows a mean time complexity of 3.896 and a

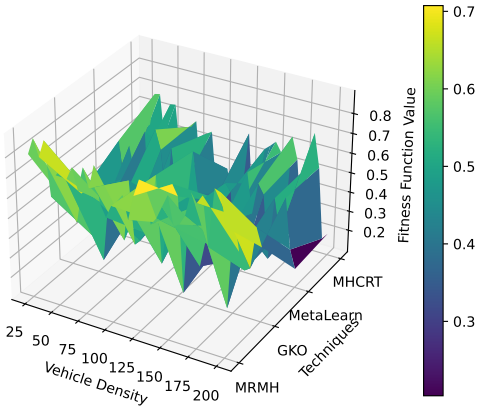


Figure 4.9: MRMH fitness function measure

higher standard deviation (0.371), indicating less consistent performance. The Friedman and Kruskal-Wallis (KW) tests further validate these findings, with MRMH showing the lowest Friedman test statistic (1.42) and a significant p-value in the KW test (2.041E-6), underscoring its superior time efficiency over methods like MetaLearn, GWOCNET, and MOCRAW. Regarding cluster stability, MRMH leads with a mean of 35.512 and a median of 39.312, indicating robust maintenance of stable communication links within a highly dynamic vehicular network. Its standard deviation (7.135) is comparable to others, reflecting reasonable variance across different scenarios. In contrast, MetaLearn shows a mean of 33.896 and a median of 36.762 in cluster stability but does not match MRMH's level. Similarly, methods like GKO and MOCRAW, despite their strengths, demonstrate lower mean and median values in cluster stability compared to MRMH.

Table 4.4: Statistical analysis of state-of-the-art and MRMH

Technique	Time Complexity					Friedman	KW	Cluster Stability		
	Mean	Best	Median	Worst	Std	Test	(p-value)	Mean	Median	Std
ACO-AODV [77]	3.896	3.894	3.184	3.956	0.371	2.98	3.081E-6	28.591	30.523	10.42
MetaLearn [132]	2.214	1.967	2.286	2.554	0.206	1.95	2.143E-8	33.896	36.762	8.785
GSO-RBVT-R [78]	4.189	3.701	3.927	4.322	0.231	2.41	3.019E-6	31.892	33.458	7.589
GKO [79]	2.266	2.098	2.101	2.414	0.312	2.09	2.321E-6	33.821	36.288	6.187
Rajput et al. [81]	2.765	2.512	2.906	3.254	0.165	2.05	2.213E-4	32.942	35.412	8.431
GWOCNET [123]	2.041	1.684	1.984	2.465	0.212	2.32	2.318E-6	32.593	34.311	9.361
MHCRT [82]	3.726	3.611	3.789	3.911	0.192	2.28	3.038E-6	30.686	34.003	7.442
MOCRAW [80]	2.131	2.018	2.138	2.346	0.244	2.02	2.189E-6	32.586	34.981	6.812
MRMH	1.376	1.002	1.673	1.821	0.172	1.42	2.041E-6	35.512	39.312	7.135

Computation Time Complexity

The MRMH algorithm's time complexity is governed by its nested loops and their internal operations. Each vehicle in the vehicular space VS necessitates position calculations for the alpha, beta, delta, and omega wolves, resulting in a complexity of $O(n)$, where n is the vehicle count. The fitness function for each vehicle also has a complexity of $O(n)$. Coefficient updates, being equation-based, have a constant complexity of $O(1)$. The inner loop's of SQP operation exhibits a complexity of $O(p \log p)$, with p denoting the state space size. As both loops iterate up to $Itr_{\max}$ times, the algorithm's cumulative complexity is $O(m \times (n + p \log p))$,

where m represents the iteration limit. Thus, the MRMH complexity is shaped by the vehicle count n, the SQP complexity $O(p \log p)$, and iteration count m, combining linear and logarithmic components.

Concluding Remarks

This research provides a comprehensive examination of heuristic-based optimization methods specifically for VANETs. Through analysis of algorithmic foundations and practical implementations, this chapter emphasizes the critical role of heuristic optimizations in enhancing network performance and managing the dynamic complexities inherent in vehicular communications. Initially, MetaLearn integrated ML techniques with heuristic-based optimizations to enhance routing efficacy in VANETs. By employing GWO alongside TDL, MetaLearn dynamically adjusts routing decisions based on real-time network conditions. This approach mitigates common issues in VANETs, such as congestion and link latency, by enabling the system to adapt promptly to environmental changes. The use of Q-learning facilitates optimal route discovery by continuously updating Q-values, which reflect the likelihood of successful data transmission across the network. MetaLearn's innovative use of RL and its implementation underscore its potential for real-time adaptability and robust communication management within highly dynamic vehicular networks.

On the other hand, MRMH advances the state-of-the-art by hybridizing GWO with SQP. This synergy leverages GWO's global search capabilities and SQP's local optimization strengths, addressing both exploration and exploitation phases effectively. The introduction of a nonlinear decay mechanism and refined weighted distance formulations enhances the adaptability of routing protocols to the volatile nature of VANETs. MRMH's fitness functions, which incorporate critical VANET metrics such as inter-cluster distance and vehicle orientation, significantly improve clustering processes and routing accuracy. The methodology emphasizes real-time analysis and dynamic response to network conditions, facilitated by OBUs and RSUs, ensuring robust performance under varying traffic scenarios. Experimental validation conducted through high-fidelity simulations utilizing OpenStreetMap data and SUMO tools has demonstrated that MRMH consistently outperforms extant state-of-the-art techniques, including MetaLearn, ACO-AODV, and GWOCNET. This performance is evidenced across various metrics such as transmission delay, PDR, and throughput under diverse traffic conditions and vehicle speeds. The advanced fitness functions developed in this research, which consider inter-cluster distance and vehicle orientation, have been pivotal in optimizing clustering processes and enhancing routing accuracy. A core contribution of this research is the development of a dynamic, real-time adaptable routing mechanism. The proposed system can rapidly adapt to changing network conditions, thereby maintaining robust and efficient communication. The novel nonlinear decay mechanism and refined weighted distance approach introduced in MRMH further enhance the adaptability and responsiveness of the routing protocols to the inherent volatility of VANETs. Moreover, this research addresses the computational complexities associated with NP-complete problems in vehicular routing. The proposed hybrid metaheuristic approach not only mitigates computational overhead but also ensures efficient resource utilization, making it highly suitable for real-world applications in ITS.