Autonomous Intersection Management Platform (AIMP): Metaheuristic Optimization for Traffic Scheduling

***Abstract*—Autonomous Intersection Management (AIM) enables signal-free coordination of Connected and Automated Vehicles (CAVs) to reduce delay while maintaining safety. This paper presents a priority-aware AIM framework formulated as a mixed discrete–continuous optimization problem based on a conflict-point abstraction, where decision variables include vehicle passing order and speed assignments. Multiple metaheuristic algorithms are implemented and compared, including Simulated Annealing, Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization, Dragonfly Algorithm, and a proposed Hybrid ACO–PSO method. Experimental results demonstrate that explicitly decomposing permutation and speed optimization improves convergence stability, robustness, and computational efficiency, making the hybrid approach a balanced solution for priority-aware autonomous intersection scheduling.**

I. Introduction

Urban congestion represents one of the most critical challenges in modern transportation systems. According to the INRIX Global Traffic Scorecard, traffic congestion generated an economic cost of $461 billion in 2017 considering only the US, UK, and Germany [1] . Intersections are responsible for a disproportionate share of these losses studies indicate that nearly one quarter of fatal collisions occur at intersections despite their limited spatial footprint within road networks [1]. Furthermore, intersection-related delays significantly increase fuel consumption and emissions, aggravating both environmental and economic impacts.

With the emergence of Connected and Automated Vehicles (CAVs), Autonomous Intersection Management (AIM) has been proposed as a signal-free alternative to conventional traffic lights. Traditional signal control introduces fixed-cycle delays that often result in unnecessary stopping and acceleration phases, reducing efficiency and increasing energy consumption [2]. Planning-based signal-free strategies have demonstrated superior performance compared to FIFO-based methods, particularly under high traffic volumes, although at significantly higher computational cost [3].

Recent advances show measurable efficiency gains through optimization-based approaches. For example, Genetic Algorithm-based intersection optimization achieved throughput improvements between 9.21% and 36.98% compared to traditional traffic control methods [1].

Similarly, bus-priority scheduling models for signal-free intersections reduced average per-capita delay by 18.95% compared to non-priority scenarios [4]. Distributed MILP-based scheduling demonstrated significant reductions in intersection delay and number of stops compared to conventional signalized control [5].

Despite these promising improvements, AIM remains computationally challenging. As highlighted in [3], the planning-based crossing sequence optimization grows exponentially with the number of vehicles. Moreover, AIM inherently requires simultaneous optimization of two tightly coupled decision spaces: the discrete combinatorial scheduling of vehicle crossing order and the continuous optimization of motion trajectories [6]. This mixed discrete–continuous, nonlinear, and combinatorial structure motivates the development of scalable metaheuristic optimization frameworks.

Despite these contributions, three key limitations remain. First, many existing studies optimize either discrete vehicle ordering or continuous trajectory variables separately, rather than addressing the fully coupled mixed decision space. Second, fair comparative evaluations across multiple metaheuristics under a unified feasibility and evaluation framework are limited. Third, priority-aware scheduling formulations are frequently embedded within algorithm-specific architectures, complicating objective comparison.

To address these challenges, this paper proposes the Autonomous Intersection Management Platform (AIMP), a unified and priority-aware metaheuristic optimization framework. The intersection is modeled using a conflict-point abstraction, and the decision variables consist of a global vehicle passing permutation Π and a vector of constant vehicle speeds $\mathbf{v}$. Vehicle delays are computed relative to free-flow traversal, and the objective function minimizes a weighted sum of emergency and total delays. To efficiently solve this problem, a hybrid ACO–PSO strategy is introduced. Ant Colony Optimization is first employed to explore the combinatorial permutation space and identify high-quality vehicle passing orders. Subsequently, Particle Swarm Optimization refines continuous speed assignments while preserving the optimized permutation structure. This sequential hybridization exploits the discrete exploration strength of ACO and the rapid continuous convergence properties of PSO. All metaheuristic algorithms operate within a unified scheduling decoder that enforces feasibility constraints and ensures consistent objective evaluation, thereby enabling fair and systematic comparison across search strategies.

The remainder of this paper is organized as follows. Section II presents the formal problem formulation, including decision variables and constraints.

Section III describes the proposed methodology and scheduling decoder. Section IV details the implementation and system architecture. Section V presents experimental results and comparative performance analysis. Section VI discusses key findings. Finally, Section VII concludes the paper and outlines directions for future work.

TABLE I: Feature Comparison of Existing AIM Approaches and the Proposed Hybrid Framework

Method	Speed	Acceleration	Sequence	Priority	Main Comments
Convex Optimal Control [3]	✓	✓	✗	✗	Optimizes continuous trajectories no crossing order optimization.
Genetic Algorithm [1]	✗	✗	✓	✗	Optimizes vehicle passing cadence focuses on throughput.
MILP Scheduling [5]	✗	✗	✓	✗	primarily discrete scheduling.
Priority-Based AIM [7]	✓	✓	✗	✓	Multi-objective trajectory optimization does not jointly optimize global crossing sequence.
Double-Layer PSO [4]	✗	✗	✓	✓	Entry-time optimization with bus priority continuous refinement limited.
Proposed Hybrid ACO–PSO (AIMP)	✓	✗	✓	✓	Joint mixed discrete–continuous optimization; ACO explores permutation space, PSO refines speeds.

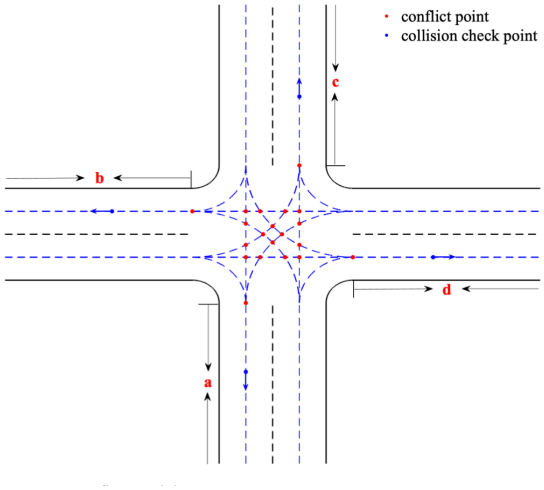


Fig. 1: Fig. 1: Intersection layout with four approaches, North, South, East and West. Red points denote conflict points between each path. Each manouver's path is shown by the dotted lines.

II. Problem Formulation

We consider a priority-aware autonomous intersection management problem at an unsignalized four-way intersection. Vehicles approach from four predefined directions North, South, East, and West and execute fixed maneuvers straight, left, or right. The intersection is modeled using a conflict-point abstraction, where each conflict point represents a spatial resource that cannot be occupied simultaneously by more than one vehicle. The case study examined in this work consists of a single isolated intersection with 16 internal conflict points generated by the combination of all admissible maneuvers. Both regular and emergency vehicles are included in the traffic stream. Experimental scenarios evaluate varying traffic densities while maintaining identical geometric and safety parameters to ensure fair comparison across optimization methods. The objective of the proposed formulation is to minimize a weighted sum of vehicle delays while accounting for priority vehicles. For each vehicle i, the delay is defined as the difference between its scheduled exit time from the intersection and its corresponding free-flow exit time, assuming traversal at maximum allowable speed without conflicts. Let $E \subset V$ denote the subset of emergency vehicles and V the full vehicle set. The optimization problem is formulated as

$$\min_{\Pi,\mathbf{v}} f = \alpha \sum_{i \in E} \text{delay}_i + \beta \sum_{i \in V} \text{delay}_i, \tag{1}$$

where α and β are weighting coefficients that regulate the relative importance of emergency vehicle delay versus total network delay. This formulation allows the system to prioritize emergency vehicles without completely neglecting overall intersection efficiency.

The decision variables consist of both discrete and continuous components. The discrete component is a global vehicle passing order represented by a permutation Π of the vehicle set V, which determines the sequence in which vehicles access shared conflict points.

The continuous component is a vector of constant traversal speeds $\mathbf{v} = [v_1, v_2, \ldots, v_N]$, where each vehicle speed is bounded within admissible limits $v_{\min} \leq v_i \leq v_{\max}$. This mixed discrete–continuous structure captures both the combinatorial scheduling problem and the continuous motion control aspect of the intersection crossing process.

Given a candidate solution $(\Pi, \mathbf{v})$, a deterministic scheduling procedure computes the entry times of each vehicle at every conflict point along its predefined path. The formulation enforces several constraints to guarantee safety and physical feasibility. First, an earliest arrival constraint ensures that no vehicle enters the intersection before reaching its first conflict point based on its initial state. Second, mutual exclusion constraints prevent simultaneous occupation of the same conflict point by enforcing a minimum temporal headway between any two vehicles sharing that point. Third, path consistency constraints ensure that each vehicle respects the ordered sequence of conflict points along its trajectory, accounting for travel time between consecutive points based on its assigned speed. Finally, same-approach queue precedence constraints preserve the relative order of vehicles arriving from the same lane to avoid unrealistic overtaking within the storage zone.

The formulation relies on several modeling assumptions to maintain tractability while preserving realism. All vehicles are assumed to be fully connected and automated, with negligible communication delay. Vehicle motion inside the control

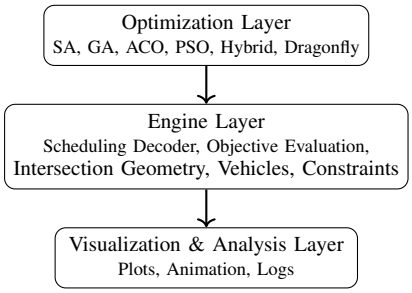


Fig. 2: Layered architecture of the proposed framework. All optimization algorithms interact with a shared scheduling decoder and evaluation engine.

zone is simplified to constant-speed traversal between conflict points, and lateral dynamics are not explicitly modeled. Overtaking within the same approach lane is prohibited. Intersection geometry and maneuver paths are assumed known a priori, and all vehicles strictly comply with assigned schedules. Under these assumptions, the resulting optimization problem is a nonlinear mixed discrete–continuous problem with combinatorial complexity, motivating the use of metaheuristic optimization methods described in the subsequent section.

III. Methodology

The proposed framework is designed as a unified mixed discrete–continuous optimization environment for priority-aware autonomous intersection management. The methodology decomposes the problem into two tightly coupled components: a deterministic scheduling decoder responsible for feasibility enforcement and objective evaluation, and a stochastic metaheuristic optimization layer responsible for exploring the search space. All optimization algorithms operate on the same decision variables , while the decoder enforces the constraints defined in Section II. This separation ensures consistency, modularity, and fair comparison across different search strategies.

A. Overall System Architecture

The system follows a layered architecture composed of three logically separated but tightly connected components: the Optimization Layer, the Engine Layer, and the Visualization and Analysis Layer. The Optimization Layer generates candidate solutions, the Engine Layer evaluates them using the scheduling decoder, and the Visualization Layer provides runtime monitoring and offline analysis. This architecture guarantees that all algorithms interact with the same modeling and evaluation core, isolating algorithmic performance from implementation bias.

B. Engine Layer and Scheduling Decoder

The Engine Layer encapsulates the full problem definition, including the conflict-point abstraction, vehicle paths, safety constraints, and delay computation. It is completely independent of the optimization algorithms. Each candidate solution $(\Pi, \mathbf{v})$ is evaluated through a deterministic scheduling decoder that constructs a collision-free traversal schedule.

Algorithm 1 Scheduling Decoder

Require: Vehicle order Π, speed vector $\mathbf{v}$, conflict-point set $\mathcal{P}$, paths $\mathcal{P}_i$, headways τ_p
Ensure: Vehicle delays $\{\mathrm{delay}_i\}_{i\in\mathcal{V}}$
Initialize availability clocks $A[p] \leftarrow 0$
Compute earliest arrival times t_i^{ear}
Initialize path indices $k_i \leftarrow 1$
while unfinished vehicles exist **do**
  for each vehicle i in order Π **do**
    if $k_i > |\mathcal{P}_i|$ **then**
      continue
    end if
    $p \leftarrow \mathcal{P}_i[k_i]$
    Compute reachable time t_{reach}
    $t_{i,p} \leftarrow \max(t_{\mathrm{reach}}, A[p])$
    $A[p] \leftarrow t_{i,p} + \tau_p$
    $k_i \leftarrow k_i + 1$
  end for
end while
Compute exit times and delays
return $\{\mathrm{delay}_i\}$

The decoder enforces earliest arrival constraints, conflict-point mutual exclusion, path consistency, and same-approach queue precedence. Conflict points are treated as shared temporal resources with minimum headway constraints. Vehicle progression through the intersection is simulated deterministically based on the assigned permutation and constant speed values. The complete decoding procedure is presented in Algorithm 1. This decoder is the sole mechanism for feasibility enforcement and objective computation, ensuring strict comparability across metaheuristics.

C. Metaheuristic Optimization Layer

The Optimization Layer searches the mixed discrete–continuous space defined by the permutation Π and speed vector $\mathbf{v}$. All algorithms share the same solution representation and are evaluated exclusively through the scheduling decoder.

Simulated Annealing (SA) was implemented as a stochastic single-solution baseline. It iteratively perturbs both permutation and speed components while accepting or rejecting candidate solutions according to a temperature-dependent probabilistic rule. Although computationally simple, SA exhibits slow convergence in highly combinatorial spaces.

The Genetic Algorithm (GA) was implemented as a population-based evolutionary strategy. Candidate solutions are encoded as permutations combined with speed vectors. Tournament selection, permutation-specific crossover, and independent mutation operators are applied to evolve the population.

GA improves exploration capability compared to SA but may experience premature convergence when the search landscape becomes highly constrained.

Ant Colony Optimization (ACO) was employed to target the discrete permutation space. Artificial ants construct vehicle sequences probabilistically using pheromone trails that encode historical solution quality. Pheromone updates reinforce high-quality permutations over successive iterations. ACO is particularly effective for combinatorial scheduling problems.

Particle Swarm Optimization (PSO) was applied to continuous speed optimization. Each particle represents a candidate speed vector, and updates are performed using inertia, cognitive, and social components. PSO provides rapid convergence in continuous domains but is not inherently suited for discrete permutation optimization.

The Dragonfly Algorithm (DA) was implemented as a socially inspired swarm method. It models separation, alignment, cohesion, attraction, and distraction behaviors. In this work, a two-stage implementation is used, where the first stage explores the permutation space and the second stage refines speed assignments. DA provides strong exploration but at higher computational cost.

D. Proposed Hybrid ACO–PSO Approach

The main contribution of this work is a sequential Hybrid ACO–PSO strategy specifically designed to exploit the mixed discrete–continuous structure of the problem. Instead of optimizing permutation and speeds simultaneously, the hybrid method decomposes the problem into two coordinated stages.

In the first stage, Ant Colony Optimization is used to explore the combinatorial permutation space. Each ant constructs a candidate vehicle order guided by pheromone intensity and heuristic information derived from delay estimates. The scheduling decoder evaluates each permutation assuming nominal speeds, and pheromone trails are updated based on the weighted delay objective. This stage identifies high-quality structural scheduling patterns.

In the second stage, the best-performing permutation obtained from the ACO phase is fixed. Particle Swarm Optimization is then applied exclusively to optimize the continuous speed vector. Each particle represents a candidate speed assignment for the fixed order. Through iterative updates based on personal and global best solutions, PSO refines speed values while the decoder enforces all safety constraints.

This sequential decomposition leverages the strengths of both algorithms. ACO efficiently handles discrete combinatorial exploration, while PSO rapidly converges in continuous domains. By separating scheduling optimization from motion refinement, the hybrid strategy reduces search dimensionality, improves convergence stability, and achieves superior delay minimization compared to standalone approaches.

E. Visualization and Evaluation Environment

The Visualization and Analysis Layer enables both online monitoring and offline evaluation. During runtime, convergence curves and key performance indicators are displayed.

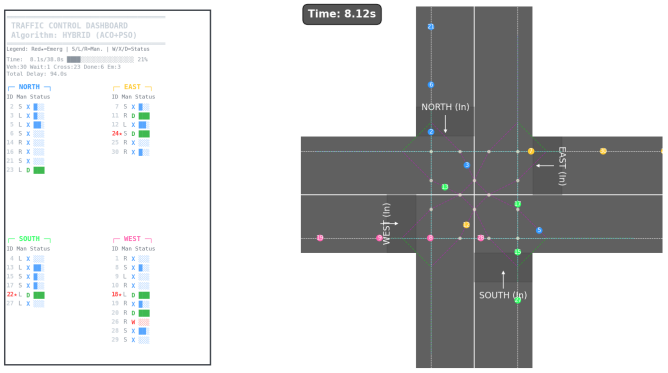


Fig. 3: Runtime visualization window used for online evaluation. The left panel displays a live dashboard summarizing optimization and scheduling metrics, while the right panel shows the intersection layout with vehicles traversing the conflict zone according to the current schedule.

TABLE II: Case Study Parameters

Parameter	Value
Number of vehicles	30
Emergency vehicles	3
Velocity range $(v_{\min}, v_{\max})$	(6, 18) m/s
Average vehicle length	4.5 m
Headway time τ	0.375 s
Safety distance	5 m
Inter-conflict distance d	5 m
Emergency delay weight α	1.0
Total delay weight β	1.0
Speed penalty coefficient	0.0
Following slack	0.1 m/s

IV. Results and Discussion

This section presents a comprehensive evaluation of the proposed priority-aware autonomous intersection management framework. All optimization algorithms were implemented in Python and executed on a MacBook equipped with an Apple M1 processor and 16 GB RAM. The development and experimental environment was based on the Visual Studio Code (VS Code) integrated development environment. All algorithms were executed sequentially under identical computational conditions to ensure fair runtime comparison.

A. Case Study Description

The case study considers a single isolated unsignalized four-way intersection modeled using a conflict-point abstraction. Vehicles approach from four directions (North, South, East, and West) and execute predefined maneuvers including straight, left-turn, and right-turn movements. The intersection contains 16 internal conflict points generated by the intersection of vehicle paths.

A total of 30 vehicles are considered in each simulation scenario. Among these, 3 vehicles are designated as emergency vehicles. All experiments were conducted under the geometric and safety parameters summarized in Table II. These parameters remain fixed across all optimization methods to ensure strict comparability.

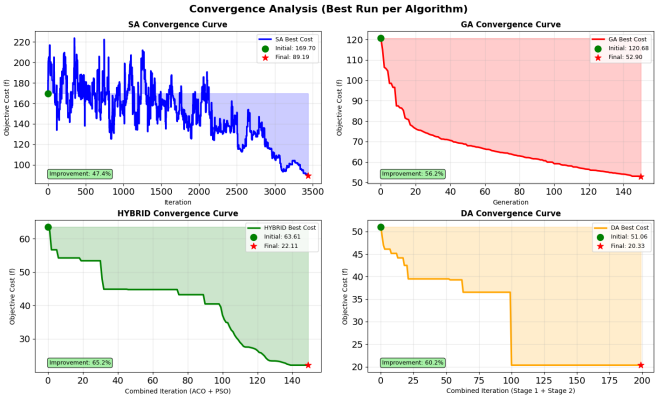


Fig. 4: Convergence behavior of the best-performing run for each algorithm.

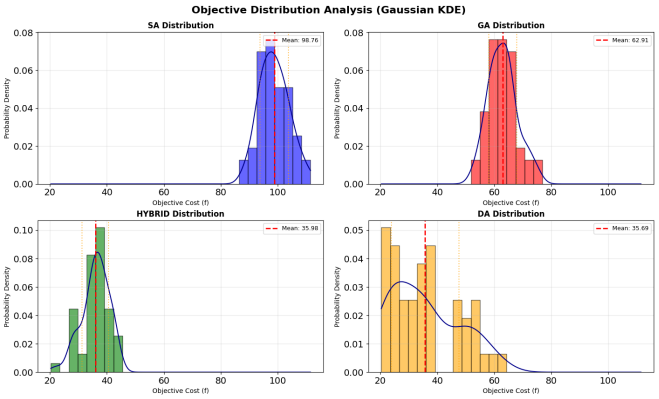


Fig. 5: Objective value distribution over 50 runs (Gaussian KDE).

B. Experimental Protocol

Each optimization algorithm was executed for 50 independent runs using randomized initial solutions. Performance was evaluated using final objective value, best achieved objective value, convergence behavior, robustness across runs, and average execution time. Since all algorithms share the same scheduling decoder and objective evaluation pipeline, differences in performance arise exclusively from the underlying search mechanisms.

C. Algorithm Hyperparameters

To ensure reproducibility and transparency, the hyperparameters used for each metaheuristic are summarized in Table III. Parameter values were selected based on literature recommendations and preliminary tuning to achieve stable convergence while maintaining fairness across algorithms.

D. Discussion

Figures 4 and 5 provide complementary insight into convergence dynamics and statistical robustness across 50 independent runs. The convergence curves highlight search behavior over iterations, while the KDE distributions quantify variability and repeatability.

Simulated Annealing improves from 169.7 to 89.2 (47.4% improvement), yet its trajectory is highly oscillatory. The stochastic acceptance of non-improving solutions promotes exploration but results in unstable convergence in the mixed permutation–continuous search space. This behavior is reflected in the wide KDE distribution (mean 98.8), confirming limited robustness and sensitivity to initialization.

The Genetic Algorithm demonstrates smoother convergence, reducing the objective from 120.7 to 52.9 (56.2%). Population-based recombination improves exploitation, producing a more concentrated distribution (mean 62.9) compared to SA. However, the gradual flattening of the convergence curve suggests occasional premature convergence, limiting its ability to consistently reach lower-cost regions.

The Dragonfly Algorithm exhibits a clear two-stage convergence pattern, ultimately achieving 20.3 (60.2% improvement). The sharp drop during later iterations corresponds to intensified exploitation after exploratory swarm dynamics. Although DA attains the best single-run result, its broader distribution (mean 35.7) indicates higher variability, reflecting sensitivity to swarm interactions and parameter tuning.

The proposed Hybrid ACO–PSO method shows the most structured convergence, decreasing from 63.6 to 22.1 (65.2% improvement). The early phase efficiently explores the discrete permutation space via ACO, while the subsequent PSO refinement stabilizes continuous speed optimization. This decomposition reduces cross-interference between discrete and continuous variables, resulting in a tightly concentrated distribution (mean 36.0) and superior repeatability.

Figures 6a, 6c, and 6b summarize peak performance, robustness, and computational cost. DA achieves the lowest best objective (20.33), closely followed by the Hybrid method (22.11), both substantially outperforming GA (52.90) and SA (88.60). However, robustness analysis in Fig. 6c reveals that the Hybrid method exhibits the narrowest interquartile range, indicating highly consistent performance across runs.

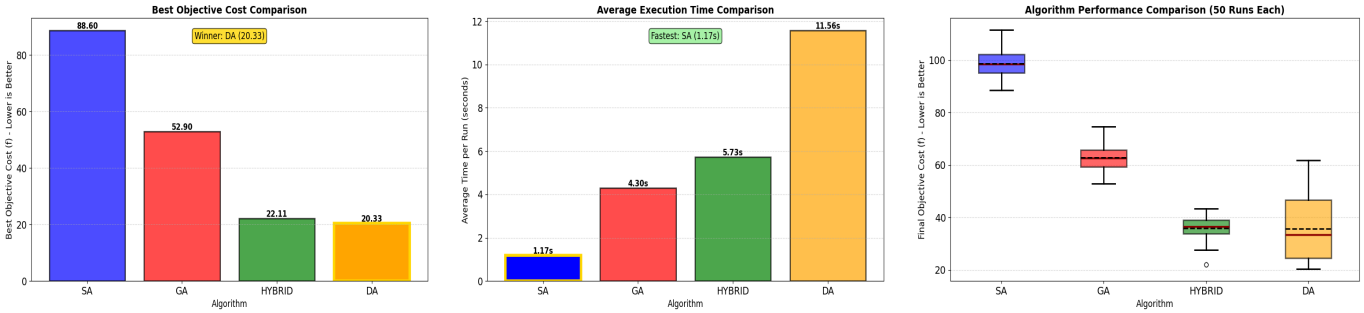
DA, while optimal in the best case, displays greater dispersion. SA and GA show higher medians and larger spreads, confirming inferior stability. In terms of runtime, SA is the fastest method (1.17s per run), but this computational efficiency comes at the expense of significantly weaker solution quality and higher variance. The Genetic Algorithm requires 4.30s per run, offering improved objective values but still showing limitations in consistently reaching low-cost regions of the search space.

The Hybrid ACO–PSO approach requires 5.73s per run, reflecting the additional computational effort of sequentially optimizing permutation and speed components. The Dragonfly Algorithm exhibits the highest runtime (11.56s per run), primarily due to its dual-stage swarm dynamics and extensive neighborhood interactions.

These results highlight a clear trade-off between optimality and computational cost. While DA achieves strong peak performance, it does so with substantial runtime overhead and increased variability. In contrast, the Hybrid approach provides a favorable compromise, delivering near-optimal solutions

TABLE III: Metaheuristic Hyperparameters

Algorithm	**Hyperparameters**
Simulated Annealing (SA)	$T_0 = 1000$, $T_{\min} = 1$, cooling rate $= 0.99$, iterations per temperature $= 5$, maximum iterations $= 10^5$
Genetic Algorithm (GA)	Population size $= 150$, generations $= 150$, elitism rate $= 0.1$, tournament size $= 5$, permutation mutation rate $= 0.2$, speed mutation rate $= 0.4$, early stopping patience $= 25$
Particle Swarm Optimization (PSO)	Swarm size $= 20$, iterations $= 100$, inertia weight $= 0.5$, cognitive coefficient $c_1 = 1.4$, social coefficient $c_2 = 1.0$, early stopping patience $= 60$
Ant Colony Optimization (ACO)	Number of ants $= 100$, iterations $= 200$, pheromone importance $\alpha = 1.0$, heuristic importance $\beta = 2.0$, evaporation rate $\rho = 0.5$, pheromone constant $Q = 500$, elitist weight $= 1.0$, early stopping patience $= 30$
Dragonfly Algorithm (DA)	Stage 1 swarm size $= 60$, Stage 2 swarm size $= 60$, iterations per stage $= 100$, separation $= 2.0$, alignment $= 2.0$, cohesion $= 2.0$, attraction $= 1.0$, distraction $= 1.0$, inertia $\in [0.1, 0.5]$
Hybrid ACO–PSO	ACO stage identical to standalone ACO; PSO stage identical to standalone PSO; sequential execution with fixed permutation during PSO refinement



(a) Best objective value

(b) Average runtime

(c) Performance distribution

with significantly lower computational demand and greater consistency across runs.

Overall, explicitly leveraging the mixed discrete–continuous structure through sequential ACO–PSO hybridization enhances convergence stability, improves reproducibility, and maintains moderate runtime complexity. The Hybrid method effectively balances exploration and exploitation, making it the most well-rounded and practically deployable solution for priority-aware autonomous intersection scheduling.

V. Conclusion

This paper presented a priority-aware autonomous intersection management framework based on a mixed discrete–continuous optimization formulation. By explicitly separating vehicle permutation optimization using ACO from continuous speed refinement using PSO, the proposed Hybrid ACO–PSO method effectively exploited the intrinsic structure of the problem. Experimental results demonstrated improved convergence stability, strong robustness across runs, competitive optimality, and a favorable trade-off between solution quality and runtime compared to standalone metaheuristics. Nevertheless, the study is limited to a single isolated intersection with fixed traffic demand and simplified constant-speed vehicle modeling under fully autonomous assumptions. Future work will extend the framework to multi-intersection networks, incorporate dynamic arrivals and acceleration-based trajectory modeling, and explore distributed or parallel implementations to enhance scalability and real-time applicability.

References

[1] L. Cruz-Piris, M. López-Carmona, and I. Marsá-Maestre, "Automated optimization of intersections using a genetic algorithm," *IEEE Access*, vol. PP, pp. 1–1, 01 2019.

[2] C. Wang, Y. Dai, and J. Xia, "A cav platoon control method for isolated intersections: Guaranteed feasible multi-objective approach with priority," *Energies*, vol. 13, p. 625, 02 2020.

[3] X. Pan, B. Chen, S. Timotheou, and S. A. Evangelou, "A convex optimal control framework for autonomous vehicle intersection crossing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, p. 163–177, Jan. 2023. [Online]. Available: http://dx.doi.org/10.1109/TITS.2022.3211272

[4] D. Wang, S. Jiang, G. Zheng, and X. Shi, "An optimal vehicle-scheduling model for signal-free intersections considering bus priority in a connected and automated vehicle environment," *Sensors*, vol. 25, p. 5438, 09 2025.

[5] F. Ashtiani, S. A. Fayazi, and A. Vahidi, "Multi-intersection traffic management for autonomous vehicles via distributed mixed integer linear programming," 06 2018.

[6] A. Abbas-Turki, Y. Mualla, N. Gaud, D. Calvaresi, W. Du, A. Lombard, M. Dridi, and A. Koukam, "Autonomous intersection management: Optimal trajectories and efficient scheduling," *Sensors*, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1509

[7] H. Zhang, R. Zhang, C. Chen, D. Duan, X. Cheng, and L. Yang, "A priority-based autonomous intersection management (aim) scheme for connected automated vehicles (cavs)," *Vehicles*, vol. 3, pp. 533–544, 08 2021.