Hindawi
Journal of Robotics
Volume 2020, Article ID 6217409, 22 pages
https://doi.org/10.1155/2020/6217409

Research Article

Scheduling-Based Optimization for Motion Coordination of Autonomous Vehicles at Multilane Intersections

Mehmet Ali Guney[1] **and Ioannis A. Raptis**[2]

[1]*Department of Mechanical Engineering, University of Massachusetts Lowell, Lowell, MA 01854-5104, USA*
[2]*Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411, USA*

Correspondence should be addressed to Ioannis A. Raptis; iraptis@ncat.edu

Received 31 December 2019; Accepted 1 February 2020; Published 31 March 2020

Academic Editor: L. Fortuna

This paper considers the motion coordination problem of autonomous vehicles in an intersection of a traffic network. The featured challenge is the design of an intersection traffic manager, in the form of a supervisory control algorithm, that regulates the motion of the autonomous vehicles in the intersection. We cast the multivehicle coordination task as an optimization problem, with a one-dimensional search-space. A model- and optimization-based heuristic method is employed to compute the control policy that results in the collision-free motion of the vehicles at the intersection and, at the same time, minimizes their delay. Our approach depends on a computation framework that makes the need for complex analytical derivations obsolete. A complete account of the computational complexity of the algorithm, parameterized by the configuration parameters of the problem, is provided. Extensive numerical simulations validate the applicability and performance of the proposed autonomous intersection traffic manager.

1. Introduction

In the last decade, significant progress has been made in integrating wireless communications, advanced sensing technologies, and computationally powerful embedded systems on-board vehicles and transportation systems. These augmenting technologies are projected to mitigate existing problems of urban traffic such as congestion, safety, reliability, and sustainability [1]. The most notable challenge in road traffic management systems is the control of traffic in junction nodes of the transportation network where roads cross or merge such as intersections, roundabouts, and onramps [2, 3]. Traffic intersections in particular, even though they constitute a small portion of the overall road transportation network, they account for a disproportionately high amount of traffic accidents (in both US [4] and EU [5]) and are predominant locations where traffic congestion aggravates place.

Advances in the fields of communication and information technologies, customized for road transport, and vehicles autonomy gave rise to Intelligent Transportation Systems (ITSs) [6], which are commissioned to improve the efficiency of existing traffic management and mobility. Present-day vehicles are equipped with a variety of sophisticated sensors suitable for situational awareness and capable of measuring with sufficient accuracy the vehicle's state, e.g., location, speed, and acceleration. At the same time, emerging technologies in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications (jointly referred to as V2X communications) allow the information exchange between powerful computing systems on-board the vehicles and the infrastructure, which are capable of real-time processing of the information collected by the traffic system and online decision-making.

The emergent integrated networked systems of autonomous, computing, and sensing nodes are brought into play to automate the traffic management of intersections. The overarching objective of this trending research field is to improve the safety and efficiency performance of intersections, compared to traditional means of traffic control

(signals, signs, and right-of-way conventions). The long-term vision is to have autonomous vehicles transporting in accident-free traffic systems. The featured module of automated intersections is the control algorithm, also referred to as Autonomous Intersection Manager (AIM) [7, 8], that is responsible for coordinating the collision-free motion of the vehicles within the intersection and also optimizes a performance metric of the traffic system. Due to the large volume of data, the scale and complexity of the problem, the multitude of factors that introduce uncertainty (wheel slippage, modeling imperfections, and packet drops to the communication system), and the processing and bandwidth limitations, the design of efficient control algorithms for autonomous intersection management is becoming a prominent challenge.

The motion coordination of vehicles in autonomous intersections manifests a unique control problem. Its complexity makes the derivation of closed-form solutions cumbersome or analytically infeasible. The problem itself is cross-disciplinary: the autonomous intersection is a multi-agent system, subjected to asynchronous communication broadcasts, where nonlinear equations govern the agents' dynamics. Traditional nonlinear systems theory is required for the design of the low-level feedback control laws that steer the vehicles based on their kinematic (or dynamic) configuration; elements from computational optimization and dynamic programming are needed to seek for an optimal control policy that minimizes the performance metrics of the optimization problem; and heuristic algorithms are requisite to efficiently search for an acceptable solution in the high-dimensional problem space. All these computational and analytical components need to be amalgamated into a single framework that will be computationally tractable and applicable to a computing system.

Seminal work on the design of decision-making algorithms for automated road systems can be traced all the way back to the 1960s in the work of Athans [9], and Levine and Athans [10], where the merging of streams of vehicles and the coordination of vehicles in a single string (platooning) were originally cast as optimal control problems. It should be noted that the intersection traffic management bears distinct differences from conventional collision avoidance applications, with the most distinctive being airspace control. The vehicles in intersections are expected to follow predefined paths; the AIM can only change the temporal parameterization of the vehicles' trajectory by adjusting their acceleration. The fact that multiple vehicles need to transport over a confined shared space, with no option of readjusting their route, makes the intersection control problem NP-hard [11, 12].

In the existing literature, the classification of autonomous intersection controllers is principally based on the architecture (concentration) of the controller. The intersection controllers are mainly classified into centralized and decentralized [3, 13–15]. Centralized methods employ a singleton decision-making unit that globally plans the motion of all vehicles in the intersection, and they are broadly categorized as reservation optimization-, model predictive control-, and queuing theory-based. A prominent work on reservation-based autonomous intersection controllers is reported in [7, 16] where the connected vehicles transmit reservation requests to the AIM. The AIM either approves or denies the request, by considering the availability of the intersection. In the case of disapproval, the driver agent slows down and sends a new reservation request with an updated arrival time. Model predictive control methods plan the optimal collision-free trajectory of each vehicle by evaluating the cost of the motion in a future time horizon [17–19]. The optimization methods achieve minimization of performance metrics such as travel time within the intersection [20–22], the length of overlapped trajectories of the vehicles [23, 24], and fuel consumption [25]. Polling policies, which fall under the subject of queuing theory, for autonomous intersections have been introduced in [26, 27], where the AIM is modeled as a server, which accommodates multiple queues. The server determines which motion direction to serve based on several criteria such as the vehicles' service time and the queue length in each intersecting road.

In contrast to centralized AIMs, the decision-making process in decentralized architectures takes place at the vehicle level, using local information exchange between communicating vehicles [3]. There has been a considerable research effort on the decentralized intersection control; proposed formulations include decentralized reservation, conflict resolution, and optimization-based control methods. In [28, 29], the centralized reservation scheme is revised with a decentralized architecture using a V2V communication protocol and ad hoc rules. The conflict resolution approaches introduce a fixed priority graph to define the motion precedence between vehicles [30, 31]. Based on the priority scheme, the motion planner optimizes the trajectory of the conflicting vehicles. The decentralized optimization methods essentially formulate the motion coordination as a multiobjective optimization problem [3]. The optimization objectives include but are not limited to the travel time, energy consumption, and collision avoidance terms [13, 32]. Distributed variants of the motion coordination problem in intersecting nodes are reported in [33, 34].

This work presents a supervisory AIM tasked to coordinate the motion of computer-controller vehicles in a multiroad and multilane intersection. At the intersection, it is expected that the vehicles can make a turn in any road that complies with the right-hand rule of bidirectional traffic, and it is assumed that nonlinear kinematic equations govern their motion. The traffic manager calculates the trajectory of each vehicle sequentially, in a first-come first-served order, and outputs the control policy of each vehicle in the form of a linear velocity profile. The AIM is designed such that the vehicles enter and cross the shared region of the intersection with their maximum velocity, improving the intersection's throughput.

The multivehicle coordination control problem is cast as an optimization search with nonlinear constraints, which seeks each vehicle's optimal arrival time to the shared—by all vehicles—region of the intersection. This is accomplished by uniquely mapping the arrival time to the entire control

policy (velocity profile) of the vehicle, making the arrival time the only optimization variable. The constraints of the optimization problem encapsulate the safety requirements of the intersection system and the kinematic bounds of the vehicles. A constraint handling method converts the constrained optimization problem into an unconstrained one. Subsequently, the unconstrained formalism is suitably processed by a metaheuristic optimization technique that employs multiple moving candidate solutions (called particles) that herd the search-space seeking for the optimal solution to the optimization problem. To aid the optimization search process, we analytically limit the search-space to an interval where a solution is guaranteed to reside. The search process is further assisted by a preliminary procedure that handpicks candidate solutions for which the trajectory of the vehicle is unreserved in the shared region of the intersection. These candidate solutions become initial values of the optimization procedure and improve its performance by initiating the optimization search with partially feasible trajectories.

The main contribution of this paper is the schedule-driven optimization formulation of the motion coordination problem of computer-controlled vehicles at a traffic intersection. To solve the problem, we introduce model-based heuristics, with well-defined computational complexity bounds, that can effectively explore the search-space of the complex monolithic optimization problem and generate control policies that satisfy the intersection system's performance metrics and the vehicles' safety requirements. We employ a metaheuristic optimization search formulation that conveniently captures the safety and motion constraints of the AIM problem and can be straightforwardly implemented to a computer program, in contrast to standard, analytically laborious nonlinear programming methods. In principle, metaheuristic optimization algorithms are capable of carrying out more robust searches than conventional optimization methods when the search-space is high dimensional, discontinuous, and multimodal [35]. Our approach relies mainly on computational tools that eliminate the need for rigorous, analytically derived optimal control formulations while guaranteeing performance. Towards this end, we provide a fundamental understanding of the impact that the geometric layout of the intersection and the configuration parameters of the algorithm have on the overall complexity of the AIM. Furthermore, we show how scheduling information, which pertains to the spatiotemporal availability of the intersection's shared region, can be gracefully integrated into the computational optimization algorithm that was developed for the multivehicle coordination task. The constraint handling method retains the information of infeasible solutions instead of discarding them, facilitating the search process of the metaheuristic algorithm employed to solve the optimization problem. Finally, detailed pseudocodes outline the implementation of the proposed AIM rigorously.

This paper is organized as follows. Section 2 provides a formal description of the control problem that is addressed in this paper, in particular, the model of the intersection, a description of the constrained optimization problem that represents the intersection control task, and the outline of the proposed solution. The preliminary feasibility search procedure of candidate solutions, which correspond to unreserved trajectories in the shared region of the intersection, is given in Section 3.1. A detailed exposition of the metaheuristic optimization method that is applied to the autonomous intersection problem is provided in Section 3.2. Section 3.3 describes the penalty method, which is used to convert the constrained optimization problem into an unconstrained one, and is followed by Section 3.4 that gives insight on how to quantify the severity of constraint violations in the penalty method. The computational complexity of the proposed AIM is discussed in Section 4. The performance of the proposed AIM is assessed in Section 5 via numerical simulations. Concluding remarks are given in Section 6.

2. Problem Statement

2.1. Model of the Traffic System. This paper focuses on the motion coordination problem of autonomous vehicles in a traffic intersection. The traffic system is denoted by the set of vehicles:

$$\{A_i(j,k):\ j \in \mathcal{N}_R, k \in \mathcal{N}_L\}, \tag{1}$$

where t is the discrete time variable; $i \in \mathbb{Z}^+$ is the index of the incoming vehicle ($i \longrightarrow \infty$ as $t \longrightarrow \infty$); j and k are indexes designating the entry road and lane of A_i to the intersection, respectively; and $\mathcal{N}_R$ and $\mathcal{N}_L$ are the index sets of roads and lanes in a road, respectively. The road index set is defined as $\mathcal{N}_R = \{1, \ldots, N_R\}$, where $N_R \in \{2, 4\}$ is the total number of parallel and mutually orthogonal intersecting roads, and the lane index set is defined as $\mathcal{N}_L = \{1, \ldots, N_L\}$, where N_L is the total number of lanes in every road. For simplicity, it is assumed that all the intersecting roads have equal number of lanes. The region where the roads overlap is referred to as the *intersection region* (IR), and the segments of the roads of length L, preceding the IR, are referred to as the *control regions* (CRs) of the intersection. The set $\mathcal{N}_{j,k}(t)$, with $j \in \mathcal{N}_R$ and $k \in \mathcal{N}_L$, denotes the collection of vehicles that transport within the CR of road j and lane k at the time instant t.

The planar 2D motion of vehicle $A_i(j,k)$ is described by the pose column vector $c_i = [x_i; y_i; \psi_i]$, where x_i and y_i denote the Cartesian coordinates of A_i's front bumper with respect to the intersection-fixed frame $\mathcal{F}_\mathcal{I}$, and ψ_i its heading. The heading angle is the relative angular displacement of $\mathcal{F}_\mathcal{I}$ with respect to a vehicle-fixed frame $\mathcal{F}_i$ that is rigidly attached to $A_i(j,k)$. When $\psi_i = 0$, the two frames ($\mathcal{F}_\mathcal{I}$ and $\mathcal{F}_i$) are aligned. The two frames are illustrated in Figure 1. It is assumed that the autonomous vehicles have the kinematics of unicycle, or differential drive, robot. The kinematics of vehicle A_i are given by

$$\begin{aligned} \dot{x}_i &= v_i \cos\psi_i \\ \dot{y}_i &= v_i \sin\psi_i \\ \dot{\psi}_i &= \omega_i, \end{aligned} \tag{2}$$

where v_i is the linear velocity and ω_i is the angular velocity of A_i. The vehicles cannot move in reverse; therefore, the

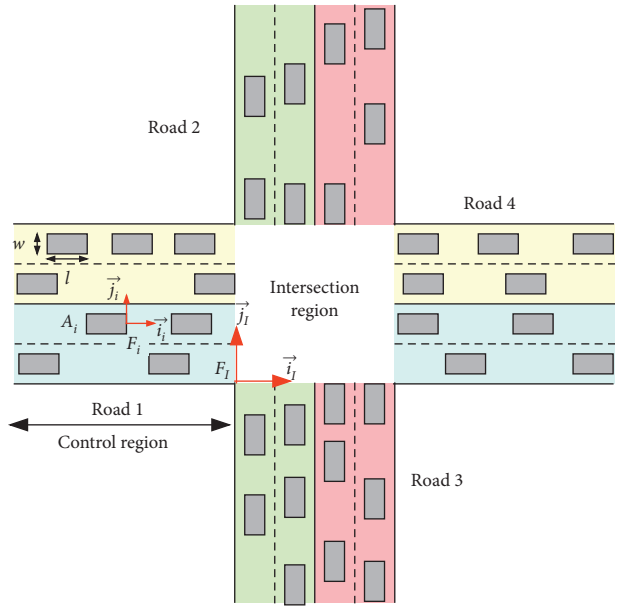

FIGURE 1: Illustration with the key features relevant to the formulation of the intersection control problem: the intersection and control regions, the geometry and dimensions of the vehicles, and the vehicle- and intersection-fixed frames.

velocity $v_i(t)$ can take values in the interval $[0, v_{\max}]$, where $v_{\max}$ is the maximum allowed velocity of the traffic system. Furthermore, their acceleration is also bounded by a positive constant $a_{\max}$; consequently, for vehicle A_i, we have $|\dot{v}_i| \le a_{\max}$. The vehicles are considered as two-dimensional rectangular rigid bodies where their width and length are denoted by w and l, respectively. Let $\mathcal{G}_i$ be the set of all points in the 2D space that are enclosed by the geometric shape of A_i. The occupancy set, $\mathcal{G}_i$, of A_i at the time instant t depends on its pose, c_i, and is defined as follows:

$$\mathcal{G}_i(t) = \left\{ \begin{array}{l} (x, y) \in \mathbb{R}^2 \,\Big|\, x_i(t) - \left(l \cdot \cos(\psi_i(t)) + w \cdot \dfrac{\sin(\psi_i(t))}{2} \right) \le x \le x_i(t) + w \cdot \dfrac{\sin(\psi_i(t))}{2}, \\ \\ \quad y_i(t) - \left(l \cdot \sin(\psi_i(t)) + w \cdot \dfrac{\cos(\psi_i(t))}{2} \right) \le y \le y_i(t) + w \cdot \dfrac{\cos(\psi_i(t))}{2} \end{array} \right\}. \tag{3}$$

Let $\mathcal{X}_i(t) = (x_i(t), y_i(t)) \in \mathbb{R}^2$ be the position coordinates of the front bumper at time t of the vehicle A_i. It is assumed that the vehicle A_i enters the CR at time $\underline{t}_i$ with the speed $\underline{v}_i = v_{\max}$. The control objective is to determine the *arrival time* $\overline{t}_i$ of A_i at the IR such that it enters the region with the speed $\overline{v}_i = v_{\max}$. The vehicles cross the IR with a constant linear velocity $v_{\max}$. Incoming vehicles to the IR can move straight or make a turn that is compatible with the right-hand convention of bidirectional traffic. The final direction of vehicle A_i is represented by the parameter $d_i \in \{0, 1, 2\}$. When $d_i = 0$, the vehicle retains its direction, else it will make a turn (left, $d_i = 1$, or right, $d_i = 2$) to one of the intersecting roads.

The crossing lanes in the IR form a square grid array. Each cell of the array is uniquely identified by its row and column indexes. The grid has $c = (N_R/2) \cdot N_L$ columns and $r = (N_R/2) \cdot N_L$ rows. Let $\mathscr{C} = \{1, \ldots, c\}$ and $\mathscr{R} = \{1, \ldots, r\}$ denote the column and row sets, respectively. A cell entry at the intersection located at the $I \in \mathscr{R}$ row and $J \in \mathscr{C}$ column is denoted by $C_{I,J}$. The edge of square cells is denoted by e and is equal to the lane's width. The enumeration order of the IR's cells is shown in Figure 2. The decomposition of IR to cells will be used later on to determine the occupancy of the IR by the incoming vehicles.

2.2. The Optimization Problem. We formulate the intersection control task, that is, the effective and collision-free multivehicle motion coordination at the intersection, as an optimization search for which its solution satisfies the specifications of the autonomous intersection in terms of performance and safety.

To begin with, an optimal traffic flow, which maximizes the intersection throughput, is achieved by minimizing the sojourn time of the vehicles at the intersection [36]. As mentioned earlier, the vehicles traverse the IR with a constant linear velocity $v_{\max}$; thus, the only delay occurs within the CR. The time it takes for A_i to travel the CR (of length L) is $t_d = \overline{t}_i - \underline{t}_i$, and A_i experiences a delay when $t_d > t_s$, where $t_s = (L/v_{\max})$. More precisely, t_s is the minimal time

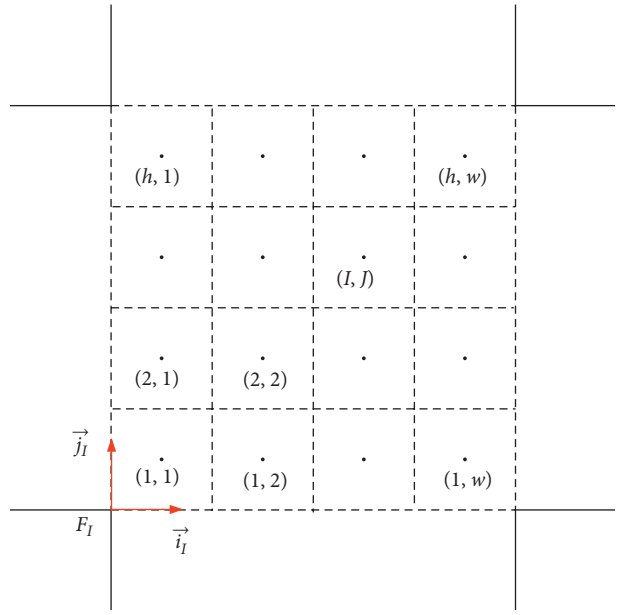

Figure 2: Grid array of the IR and its double index ordering. The enumeration of the array starts from the lower left cell of the grid array.

duration that takes a vehicle to traverse the CR with its maximum velocity, $v_{\max}$. With the following definition, we manifest the function that needs to be minimized in order to reduce the sojourn time of a vehicle A_i at the intersection.

Definition 1. The delay of vehicle A_i is a function of the time difference $\overline{t}_i - \underline{t}_i$ and is defined as follows:

$$f(\overline{t}_i, \underline{t}_i) = \overline{t}_i - \underline{t}_i - t_s. \tag{4}$$

Obviously, from the above, if there is no delay for A_i at the CR, i.e., $t_d = t_s$, then $f(\overline{t}_i, \underline{t}_i) = 0$. The vehicles are further expected to transport the intersection without colliding. The following definition states the safety requirement for the collision-free motion of the vehicles within the intersection.

Definition 2. The intersection system is safe at time t, if there are no collisions between any two vehicles $A_i(j,k)$ and $A_p(m,n)$, i.e., $\mathcal{G}_i(t) \cap \mathcal{G}_p(t) = \varnothing$, where $j, m \in \mathcal{N}_R$; $k, n \in \mathcal{N}_L$; and $p \neq i$.

Apart from the safety condition, the vehicles are subjected to kinematic constraints. In particular, the vehicles enter and exit the CR with their maximum speed, $v_{\max}$, while their velocity and acceleration in the CR should satisfy certain bounds ($|v_i(t)| \leq v_{\max}$ and $|\dot{v}_i(t)| \leq a_{\max}$ for every t in $[\underline{t}_i \overline{t}_i]$). We would like to have the arrival time as the only optimization variable and also simplify the design of the vehicles' velocity profiles in the CR. Thus, we parameterize the velocity function of A_i in the CR with its arrival time, $\underline{t}_i$, and assign a one-to-one mapping between the entry and arrival times, $\underline{t}_i$ and $\overline{t}_i$, respectively, and $v_i(t)$. To emphasize the dependency of the $v_i(t)$ function in $\underline{t}_i$ and $\overline{t}_i$, we write the velocity of A_i as $v_i(t, \underline{t}_i, \overline{t}_i)$.

Based on the above definitions—pertaining to the autonomous intersection's performance (Definition 1) and the vehicles' safety (Definition 2) and kinematic constraints—we formulate the motion coordination optimization problem for an arbitrary vehicle A_i entering the autonomous traffic intersection as follows:

$$\text{minimize } f(\overline{t}_i, \underline{t}_i), \tag{5}$$

subject to

$$\begin{aligned}
&\mathcal{G}_i(t) \cap \mathcal{G}_p(t) = \varnothing, \quad \forall t \in [\overline{t}_i, \overline{t}_i + \tau_i] \text{ and } p \neq i;\\
&\left\| \mathcal{X}_i(t) - \mathcal{X}_p(t) \right\| - l \leq 0, \quad \forall t \in [\underline{t}_i, \overline{t}_i] \text{ where } A_i(j,k), A_p(j,k) \in \mathcal{N}_{j,k}(t)\\
&\text{with } j \in \mathcal{N}_R, k \in \mathcal{N}_L, \text{and } i \neq p;\\
&0 < v_i(t, \underline{t}_i, \overline{t}_i) \leq v_{\max}, \quad \forall t \in [\underline{t}_i, \overline{t}_i] \text{ with } \underline{v}_i(\underline{t}_i) = \overline{v}_i(\overline{t}_i) = v_{\max};\\
&\left| \dot{v}_i(t, \underline{t}_i, \overline{t}_i) \right| \leq a_{\max}, \quad \forall t \in [\underline{t}_i, \overline{t}_i].
\end{aligned} \tag{6}$$

The above optimization formulation yields a constrained minimization search with a linear objective function and nonlinear constraints. The optimization variable is the arrival time $\overline{t}_i$ and the objective function, $f(\overline{t}_i, \underline{t}_i)$, is cast such

that the solution of the optimization search will minimize the delay caused by the deceleration of the vehicles in the CR. The safety constraints incorporate the nonlinear kinematic equations of the vehicles and guarantee their collision-free motion in the intersection.

2.3. Proposed Solution: Description of the Intersection Controller. The autonomous intersection utilizes a supervisory controller, which we will refer to as the *intersection controller* (IC), that regulates the motion of the vehicles that enter the intersection. The vehicles enter the intersection at random time instances and the IC determines their trajectories in a first-come first-serve manner.

When a vehicle, e.g., A_i, enters the intersection, the IC records its entry time $\underline{t}_i$. The motion control problem entails the designation of a velocity function $v_i(t)$ that satisfies the safety and kinematic constraints of A_i and minimizes its delay at the CR. However, because we parameterize $v_i(t)$ with $\underline{t}_i$ and $\overline{t}_i$, the motion control problem is conveniently converted to determining the arrival time, $\overline{t}_i$, that solves the optimization problem described in (5).

Our solution is structured as follows: a computational optimization method solves the minimization problem expressed in (5). Contrary to a conventional optimization search, the intersection control problem requires at each iteration of the search to project forward in time the vehicle's movement. This forecast is needed to examine if the safety of the vehicles at the intersection is preserved. The forward projected trajectory of A_i is compared with the trajectories of the rest of the vehicles that transport at the intersection. To make the juxtaposition between trajectories possible, a central buffer is employed that stores all the trajectories of the vehicles that move in the intersection. We refer to this buffer as the intersection's *timetable*. The computational optimization procedure reads, and updates, the entries of the timetable to examine if a candidate solution corresponds to an infeasible trajectory. At the end, the optimization procedure determines the arrival time of the vehicle at the IR, and subsequently, its velocity profile $v_i(t, \underline{t}_i, \overline{t}_i)$.

Because of the forward projection process, the optimization procedure is computationally demanding, especially when the number of candidate solutions is high. To facilitate the optimization search, we introduce a preparatory procedure that bounds the domain of the optimization problem and outputs a set of candidate solutions that correspond to trajectories with unreserved paths in the IR. These candidate solutions, which serve as a pool for initial conditions to the optimization procedure, improve the performance of the latter by initiating the search with partially feasible trajectories—in terms of safety in the IR—that may lead the search faster to an optimal solution. We show that the computational complexity of this preliminary procedure is menial compared to the complexity of the optimization search. Thus, it does not add significant computational overhead to the final algorithm; instead, it improves its performance. The block diagram of the IC is illustrated in Figure 3.

3. Description of the Intersection Controller

3.1. Preliminary Feasibility Search of Candidate Solutions. In this section, we show how the search-space of the optimization task is determined. The output of this preliminary processing operation is the assignment of a set of arrival times, to every incoming vehicle, that serve as initial candidate solutions for the optimization task. These candidate solutions are further refined during the optimization search. The IC schedules the arrival time of each incoming vehicle, and based on this value, it shapes the vehicle's linear velocity time profile in the CR. It is expected that each vehicle enters and exits the CR with its maximum velocity value, $v_{\max}$.

We assume that the routes of the vehicles have already been determined when they enter the CR and this information is available to the IC. For simplicity, we further assume that the vehicles do not change lanes, whether they remain on the same road or change direction by making a turn. However, the proposed IC can be easily modified to accommodate routes within the intersection that involve both direction and lane change.

The first task of the IC is to identify the intersection cells that the traversing vehicles will occupy along their motion in the IR—as mentioned earlier, the path of each vehicle in the IR is predetermined. Let $\mathcal{O}_i$ denote the ordered set of cells that the vehicle A_i occupies while it traverses the IR. Each cell $O_{I,J} \in \mathcal{O}_i$ is assigned with a unique *service time* $T_{I,J}$, representing the occupancy duration of $O_{I,J}$ from A_i. The service time $T_{I,J}$ is determined by the motion of the vehicle within the cell. Each vehicle can move straight or execute a quarter turn. If the vehicle does not change direction, the occupancy duration of the cell $O_{I,J}$ is $T_{I,J}^{\text{straight}} = (e/v_{\max})$. When changing directions, the vehicles move straight until reaching the cell of the IR that corresponds to their turning lane. During the turn maneuver, the vehicles retain the value of their linear velocity; therefore, the occupancy duration of the "turning" cells is calculated by $T_{I,J}^{\text{turn}} = (\pi \cdot R)/(2 \cdot v_{\max})$, where R is the turning radius of the vehicles ($R = e/2$). Figure 4 shows the two admissible trajectories of the vehicles (move straight and make a turn) within the IR and their respective duration.

The model of the turning trajectories within the IR, which consists of two straight paths and a quarter circle in between, is selected to illustrate, in an uncomplicated manner, the calculation of the total service time of a vehicle within the IR. The turning radius e can be increased, by expanding the width of the lanes, to reduce the angular velocity of the vehicles in the single-cell quarter turns in order to meet the comfort levels of the passengers while the vehicle is turning. Additionally, this simplified path can be substituted effortlessly by a curved trajectory that connects the entry and exit points of the turning vehicle in the IR and produces a lower angular velocity for the turning vehicle. In this case, the corresponding total service time of the vehicle has to be redetermined accordingly. The optimization and design of curvilinear paths is outside the scope of this work; we refer the interested reader to [37, 38].

Let $\mathcal{D}_i$ be the ordered set with elements the indices pairs of the cells that belong to $\mathcal{O}_i$, such that $\mathcal{D}_i = \{D_1, D_2, \ldots, D_{|\mathcal{O}_i|}\}$.

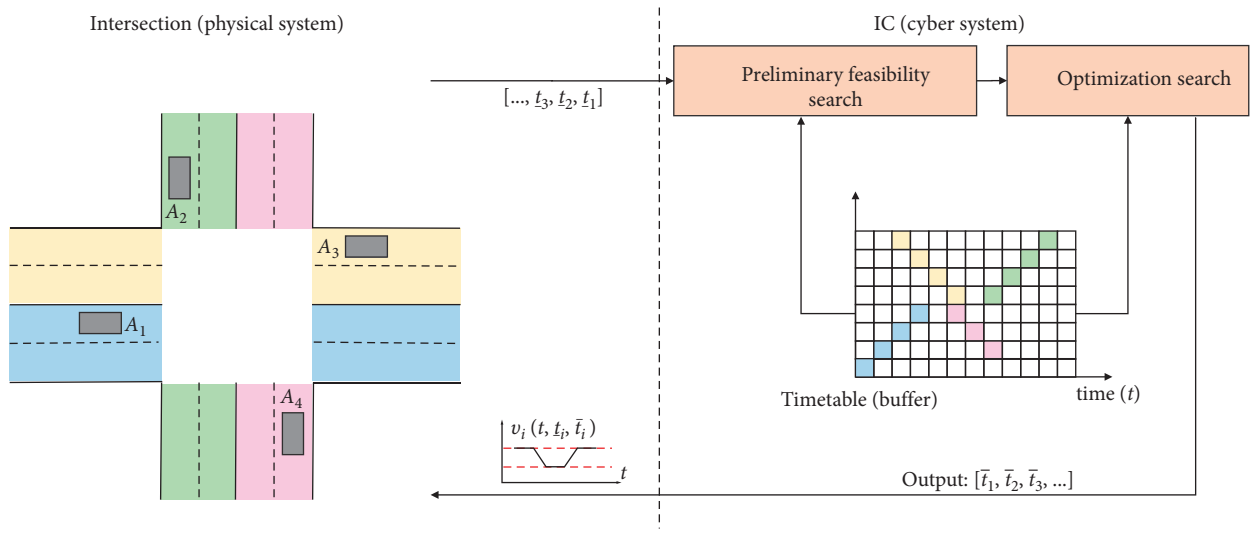

Figure 3: Block diagram of the IC. The IC receives the entry times of the vehicles when they enter the intersection. The preliminary feasibility search procedure generates a set of initial conditions for the optimization search. The output of the optimization search is the arrival time of each vehicle to the IR, which is translated into a velocity profile.

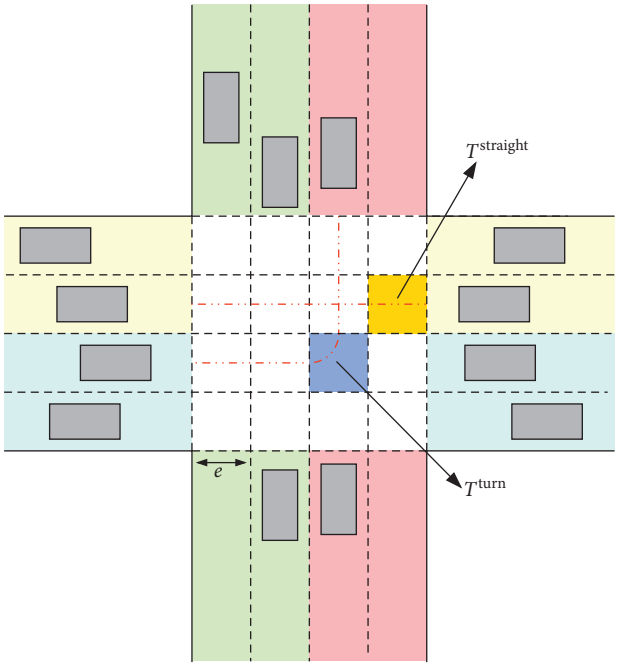

Figure 4: The two types of paths within the IR and service times of cells that correspond to straight and turning segments of paths, respectively. When a vehicle makes a turn, it maintains the same lane in its exit road with its entry road.

The total service time that is required for A_i to travel through the IR is given by

$$\tau_i = \sum_{m=1}^{|\mathcal{D}_i|} \left(T_{D_m}\right) + t_v, \tag{7}$$

where

$$t_v = \frac{l}{v_{\max}}, \tag{8}$$

is the constant duration needed for the vehicle to exit the last cell of its route in the IR.

Initially, the IC determines the time window that serves as the search-space of the optimization routine for seeking the optimal value of each vehicle's arrival time to the intersection. The duration of this time window is selected wide enough to ensure that at least one solution exists within the search-space.

A conservative value of the search window's upper bound is selected by assuming that each incoming vehicle will convey the longest possible path in the intersection while all lanes are experiencing maximum congestion. The longest possible path is illustrated in Figure 5 for a generic intersection model. The search-space's upper bound ("latest" arrival time) is calculated by the following equation:

$$\bar{t}_i^{\,l} = \underline{t}_i + \left\lceil \frac{L}{l} \right\rceil \cdot \delta_{\max} \cdot N_R \cdot N_L, \tag{9}$$

where $\lceil \cdot \rceil$ denotes the ceiling function, $\lceil L/l \rceil$ is the maximum number of vehicles that can accumulate in a single lane of the CR without colliding (maximum lane capacity), and $\delta_{\max}$ is the time duration that is required for the vehicles to convey the longest possible path in the IR (maximum service time). The maximum lane capacity and the maximum service time are multiplied by the product of roads times lanes to determine the total time that is required to serve all other vehicles before A_i arrives at the intersection—assuming that all CRs are fully congested. For an arbitrary intersection layout, the service time of the longest path in the IR, $\delta_{\max}$, is calculated by the following equation:

$$\begin{aligned} \delta_{\max} &= \left(N_R N_L - 2\right) T_{I,J}^{\text{straight}} + T_{I,J}^{\text{turn}} + \frac{l}{v_{\max}} \\ &= \frac{4\left(N_R N_L - 2\right)e + \pi e + 4l}{4v_{\max}}. \end{aligned} \tag{10}$$

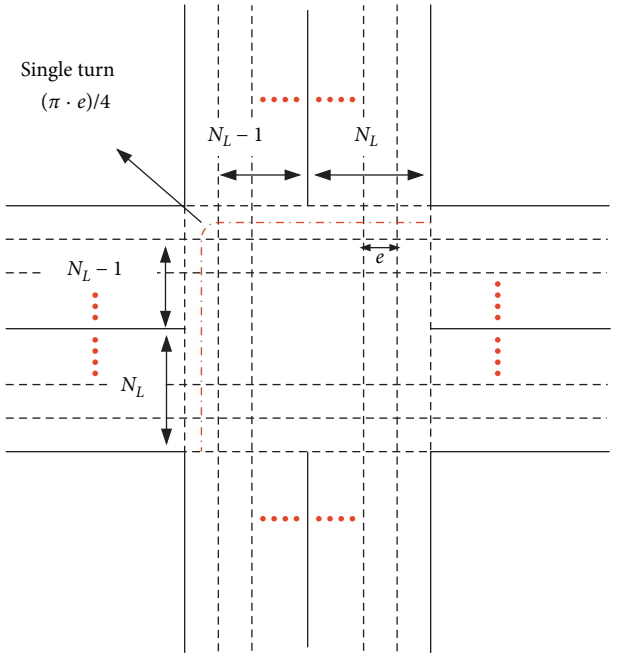

Figure 5: The longest possible path that a vehicle can travel in the IR. It is traveled by the vehicles that transport at the outermost right lane of the CR and make a left turn at the IR.

The "earliest" arrival time, $\overline{t}_i^e$, occurs when a vehicle does not experience any delay in the CR and is calculated by the following equation:

$$\overline{t}_i^e = \underline{t}_i + t_s. \tag{11}$$

Therefore, the search-space of A_i is defined as $\mathcal{S}_i = \left\{s \in \mathbb{R}^+ \mid \overline{t}_i^e \le s \le \overline{t}_i^l\right\}$. A preliminary feasible arrival time signifies that all the requested cells in $\mathcal{O}_i$ are unoccupied and available to reserve for the duration of the service times (see Figure 6). The occupancy schedule of the intersection is represented by assigning a timetable $B_{I,J}$ to the cell $O_{I,J}$ of the IR, where $B_{I,J}(t) = 0$ signals that the cell $O_{I,J}$ is available at time t, and $B_{I,J}(t) = 1$ that is occupied.

Assume that the vehicle A_i is set to traverse in the IR over a path represented by the set $\mathcal{O}_i$. The actual search-space is comprised of the discrete execution time instances of the algorithm that lie within the time interval $\mathcal{S}_i$. Denote by $S_i = \left\{s_1^i, \ldots, s_{N_i}^i \mid s_j^i \in \mathcal{S}_i\right\}$ the set of candidate arrival times of A_i, where N_i is the total number of discrete execution time instances within $\mathcal{S}_i$.

Subsequently, the IC determines the set of entrance-to-cells time instances denoted as $t^i_{\text{entrance}} = \{t^i_{D_1}, \ldots, t^i_{D_{|\mathcal{D}_i|}}\}$, where $t^i_{D_j}$ stands for the time instant that A_i enters the cell O_{D_j} with $j \in \{1, \ldots, |\mathcal{D}_i|\}$. To calculate t^i_{entrance}, we initially set $t^i_{D_1} = s^i_j$. Similarly, the set of exit-from-cell instances, denoted by $q^i_{\text{exit}} = \{q^i_{D_1}, \ldots, q^i_{D_{|\mathcal{D}_i|}}\}$, is also calculated, where $q^i_{D_j}$ marks the time instant that the vehicle exits entirely the cell O_{D_j}. Let $\mathcal{T}_j$ be the collection of all execution instances

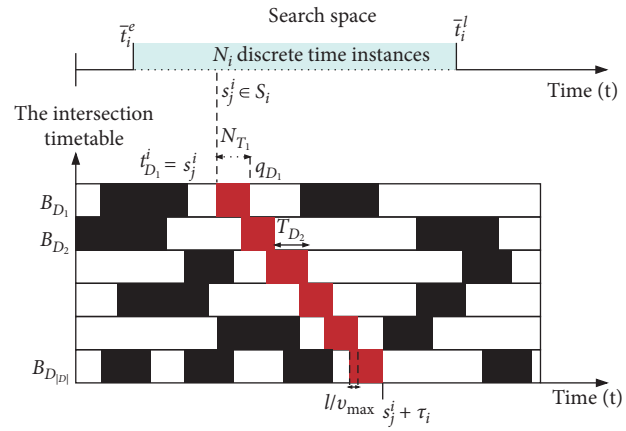

Figure 6: Graphical illustration of the intersection's reservation timetable. The timetable is a buffer that stores for each cell of the IR its occupancy information with respect to time. For an arbitrary vehicle A_i, the preliminary feasibility search identifies the elements of the set S_i that correspond to an unreserved trajectory at the IR using the timetable.

that lie within the closed interval $[t^i_{D_j}, q^i_{D_j}]$, where $j \in \{1, \ldots, |\mathcal{D}_i|\}$, and let $N_{\mathcal{T}_j} = |\mathcal{T}_j|$. The feasibility of a candidate arrival time in S_i is determined by the following function:

$$g^i(s) = \sum_{m \in \mathcal{D}_i} \sum_{n \in \mathcal{T}_n} B_{D_m}(n). \tag{12}$$

The above function is calculated sequentially for the elements of S_i; therefore, the search-space is revised as $S_i^\star = \left\{s \in S_i \mid g(s) = 0\right\}$ such that $S_i^\star \subseteq S_i$. The set $S_i^\star$ will be referred to as the *arrival times set* of A_i.

3.2. Particle Swarm Optimization. The preliminary search process of the previous section results in a queue of arrival time sets; the queue's order follows the sequence that the vehicles enter the autonomous intersection. The elements of the arrival time set $S_i^\star$ correspond to the incoming vehicle's A_i feasible arrival times to the IR. These feasible arrival times depend on the availability of the IR's timetable. Apart from the availability of the IR, a feasible arrival time solution should further satisfy the safety and kinematic constraints of the vehicle. To this end, the sets of feasible arrival times are postprocessed sequentially, in a first-come first-serve manner, by a metaheuristic optimization routine, namely, the particle swarm optimization (PSO) algorithm. The PSO seeks the suboptimal arrival time that minimizes the delay of each vehicle and, at the same time, satisfies the vehicle's safety and kinematic constraints.

The PSO algorithm was introduced in 1995 by Kennedy and Eberhart [39, 40]; it belongs to a class of metaheuristic, intelligent self-learning methods, which induce near-optimal solutions for complex optimization problems. The PSO employs a group of moving particles (swarm) to seek a solution in the search-space. The motion of the particles within the search-space is determined by their position and velocity, while a temporary memory buffer stores the fittest

(in mathematical optimization, the objective function (to be minimized or maximized) is also referred to as cost function, for minimization problems, or fitness function, for maximization problems. In the context of evolutionary programming, the terms objective and fitness functions are synonymous, regardless if the problem entails minimization or maximization. In this paper, we use an evolutionary computing method to solve a minimization problem; thus, to keep the terminology consistent for readers from both disciplines, we refer to a candidate solution as "fit" if it minimizes the objective (cost) function) location that has been explored in the search-space by the swarm [41]. At every iteration, the motion of each particle is determined based on the best-explored position of its own as well as the swarm. Therefore, the candidate solutions propagate towards the optima. Compared with other evolutionary optimization algorithms, the PSO requires less computational effort to solve complex problems [39, 42] and is relatively more robust to finding an optimal solution [43].

Due to its computational efficiency and robustness, the PSO algorithm has a broad spectrum of application areas such as robotics [44, 45], image and video analysis [46, 47], antenna design [48], sensor networks [49], signal processing [50], and scheduling [51]. To the best of the authors' knowledge, the application of the PSO algorithm to plan and coordinate the motion of autonomous vehicles at an intersection of a traffic network is novel.

The optimization problem, which the PSO is assigned to solve, involves the unconstrained minimization (or maximization) of an objective function $\phi(X)$, where $\phi\colon \mathbb{R}^n \longrightarrow \mathbb{R}$, X is a location in the search-space, and n is its dimension. The search-space is also called the *domain* of the optimization problem. The swarm of the PSO algorithm consists of N_p particles, where each particle serves as a candidate solution of the optimization problem. Let X_a and V_a, where $a \in \mathcal{P} = \{1, \ldots, N_p\}$, be the position and velocity vectors in $\mathbb{R}^n$ of the a^{th} particle in the swarm, respectively. Each particle has a *fitness value*; that is, the value of the objective function at the particle's position in the search-space, e.g., the fitness value of particle a with position X_a is $\phi(X_a)$.

The particles move around in the search-space, according to a simple set of rules that determine each particle's velocity, to find a solution X^* that minimizes the objective function, such that $\phi(X^*) \leq \phi(X)$ for all locations, X, that have been visited by the swarm, in the search-space. Each particle's movement is influenced by its own experience and the experience of the most successful particle in the swarm [42].

The initial positions of the particles are randomly assigned in the n-dimensional search-space of the optimization problem. The particle's direction of motion in the search-space is guided by two elastic forces [52]. The first force attracts the particle, with a random magnitude, towards the best (fittest) position that has been explored by the swarm, denoted by P_g. The second pulls the particle, with a random magnitude, to the fittest position that has been discovered by itself, denoted by P_a. Figure 7 depicts the two forces that determine the trajectory of a particle in the search-space. The velocity of the a^{th} particle in the swarm is calculated by the following equation:

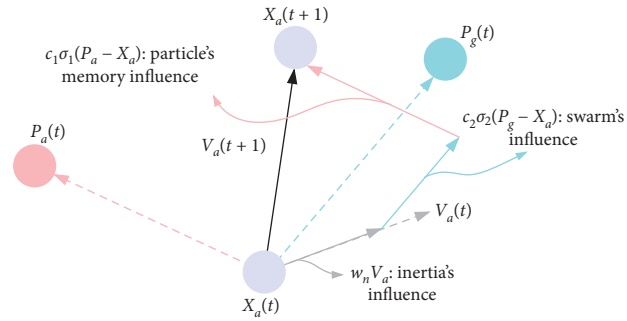

FIGURE 7: Graphical illustration of the a^{th} particle's subcomponents that form its updated velocity vector, $V_a(t+1)$, at the execution instant t of the PSO. The updated velocity is a linear combination of its current velocity, $V_a(t)$, and its relative displacements from its own and the population's fittest position values, $P_a(t)$ and $P_g(t)$, respectively.

$$V_a(t_p + 1) = w_n(t_p) \cdot V_a(t_p) + c_1 \cdot \sigma_1 \cdot (P_a - X_a(t_p)) + c_2 \cdot \sigma_2 \cdot (P_g - X_a(t_p)), \tag{13}$$

where t_p is the iteration number of the PSO algorithm, σ_1 and σ_2 are independent random variables with a uniform distribution in the closed interval $[0, 1]$, and c_1 and c_2 are the constant acceleration coefficients (the positive constants c_1 and c_2 are also referred to as the *cognitive* and *social* parameters of the swarm, respectively [42]). The acceleration coefficients encapsulate the particle's confidence to its own success and the swarm's success, and they are typically selected as $c_1 = c_2 = 2$ [42]. The search process is terminated when a prespecified number of iterations, $t_p^{\max}$, has been reached. The final solution of the optimization problem is the swarm's best discovered value, P_g.

The term w_n is the inertia weight; it balances the global and local search performance of the PSO algorithm [53]. A high value of the inertia weight yields an explorative global search while a smaller one results in a finer local search. In [53, 54], it was shown that the gradual decrease of the inertia weight, during the execution of the algorithm, notably improves the performance of the PSO. In similar fashion, we linearly decrease the inertia weight with the iteration number, t_p, to balance the search response of the swarm. With this approach, at the early iterations, the PSO tends to have a more global search ability while towards the end of the execution it tends to have a more local one. Similarly to [42], the inertia weight is calculated by the following equation:

$$w_n(t_p + 1) = w_{\max} - (w_{\max} - w_{\min}) \cdot \frac{t_p}{t_p^{\max}}, \tag{14}$$

where $w_{\max}$ and $w_{\min}$ are the prespecified maximum and minimum values of the inertia weight, respectively. The maximum inertia weight, $w_{\max}$, is generally selected as 1 while the minimum number, $w_{\min}$, is set to 0.1 (close to zero) [42].

After calculating the velocity vector V_a of particle a, its position is updated based on the following expression:

$$X_a\left(t_p+1\right)=X_a\left(t_p\right)+V_a\left(t_p+1\right). \tag{15}$$

According to (13), at each iteration, the velocity of a particle is a linear combination of its relative distances from its individual and the population's fittest position values. Thus, all particles are attracted by P_g (the population's current fittest position value), while keeping their own information (through P_a).

The individual's and population's best visited positions P_a and P_g, respectively, are updated according to the fitness values of the particles. For example, in the case of a minimization problem, if $\phi(X_a(t_p+1)) < P_g$, then the a^{th} particle's position becomes the population's best discovered value, and P_g takes the value of $X_a(t_p+1)$. The individuals' and population's best values P_a and P_g, respectively, drive iteratively the particles of the swarm towards a satisfactory (suboptimal) solution of the optimization problem. The pseudocode of the PSO procedure is outlined in Algorithm 1.

Returning to the intersection control problem, the PSO search is executed for every incoming vehicle. The initial candidate solutions, N_p in the number, for the vehicle A_i are selected randomly from the set $S_i^\star$. Thus, the position of the a^{th} particle, X_a, represents a candidate arrival time, $\overline{t}_i^a$, of A_i. As a reminder, we note that—by design—the selection of the latest arrival time, $\overline{t}_i^l$, of vehicle A_i guarantees that there exists at least one solution of the optimization problem in the search-space $[\overline{t}_i^e\ \overline{t}_i^l]$ (see Section 3.1).

Recall that the PSO is suitable for only unconstrained optimization problems. Therefore, we need to convert the original objective function—that is, the delay, $f(\overline{t}_i, \underline{t}_i)$—and its nonlinear constraints, described in (5), into a scalar function, $\phi(\overline{t}_i, \underline{t}_i)$, that encapsulates both the cost function and the constraint violations. In the following section, we show how the original constrained problem is recast into an unconstrained one, making it compatible with the PSO algorithm.

3.3. Constraint Handling Method. The intersection control problem entails the selection of each vehicle's candidate arrival time at the IR that minimizes the delay function, described in Definition 1, and satisfies the vehicle's trajectory kinematic and safety constraints. We have manifested the intersection's multivehicle collision-free motion control preconditions into a minimization problem, expressed in (5). Because of the vehicles' nonholonomic kinematics and the compound form of their trajectory profile, the constraints of the minimization problem are nonlinear and complex.

In the literature of mathematical optimization, metaheuristics are a practical and effective option for solving complex optimization problems [55]. However, as mentioned earlier, metaheuristics, and in particular PSO, are suitable for unconstrained optimization problems. Thereby, for optimization problems that include constraints, an additional mechanism is needed to handle the infeasible candidate solutions, which fail to satisfy the given constraints.

In this paper, we adopt the penalty method that methodically converts the constrained optimization problem into an unconstrained one [56, 57]. According to this formulation, an auxiliary term (penalty term) is added to the objective function of the original constrained problem. For a minimization search, the penalty term adds positive value to every *infeasible* solution—a solution is infeasible if it does not satisfy the constraints of the optimization problem—that signifies the level of violation of each constraint.

The value of the penalty function is a measure of the candidate solution's constraint violations severity. The number of constraint violations, which correspond to a candidate solution, determines the contribution of the penalty to the objective function of the unconstrained problem (for clarity (and brevity), the principal minimization problem of the intersection's motion control application, expressed in (5), will be referred to as the "constrained" optimization problem and the delay function, $f(\cdot,\cdot)$, of Definition 1 as the "constrained" objective function, whereas its unconstrained counterpart, suitable for the PSO search and described in Section 3.2, will be referred to as the "unconstrained" optimization problem and the objective function $\phi(\cdot,\cdot)$ as the "unconstrained" objective function). If for a candidate solution there is no violation of any constraint, then the penalty term is zero, and the resultant objective function becomes identical with the one of the original constrained problem. The penalty function also is a means of making use of the infeasible solutions instead of discarding them (also known as the *death penalty* method). The disadvantage of entirely discarding the infeasible solutions is that the information that they may contain will be lost.

The penalty function can be designed as static or nonstatic [58, 59]. Typically, the static penalty function is formulated as a weighted sum of the constraint violations [60]. However, this approach is highly problem-oriented because the designer needs to determine the contribution of each constraint violation into the penalty amount [55]. The nonstatic penalty function requires no explicit parameter tuning; the penalty amount that is added to each constraint violation is adaptively controlled such that the infeasible solutions gradually converge to the feasible solution region [61].

In this work, we adopt a self-adaptive penalty function that has been reported in [59]. This formulation utilizes the information and experience that is gained through the search process to regulate the penalty amount added to the infeasible solutions. The goal is to derive a new objective function that encapsulates the fitness of the original one (constrained problem) and the penalty values that quantify the constraint violations. To directly connect the constraint handling method with the PSO search formulation for the intersection control problem, the new objective function (of the unconstrained problem), $\phi(\overline{t}_i^a, \underline{t}_i)$, of an incoming vehicle A_i is parameterized by the candidate solution that corresponds to particle $a \in \mathcal{P}$ (arrival time $\overline{t}_i^a$) of the swarm and the vehicle's entrance time to the CR, $\underline{t}_i$.

The self-adaptive penalty method replaces the objective function of the constrained optimization problem with a

Procedure: PSO
Input: $S_i^\star$
Output: $\overline{t}_i$
Required: objective function
(1) Draw randomly N_p elements from the set $S_i^\star$.
(2) Initialize the particles' positions, $X_a(0)$, and fitness values, $\phi(X_a(0))$, for all $a \in \mathscr{P}$.
(3) Find P_g based on the initial fitness values of the population.
(4) $t_p \longleftarrow 1$
(5) **While** $t_p \leq t_p^{\max}$ **do**
(6) **for** $i = 1$: $\mathscr{N}_p$ **do**
(7) $V_a(t_p+1) \longleftarrow w_n \cdot V_a(t_p) + c_1 \cdot \alpha_1 \cdot (P_a - X_a(t_p)) + c_2 \cdot \alpha_2 \cdot (P_g - X_a(t_p))$ ▷ Update the velocity of the a^{th} particle.
(8) $w_n(t_p+1) \longleftarrow w_{\max} - ((w_{\max} - w_{\min})/t_p^{\max}) \cdot t_p$ ▷ Update the inertia weight term of the a^{th} particle.
(9) $X_a(t_p+1) \longleftarrow X_a(t_p) + V_a(t_p+1)$ ▷ Update the position of the a^{th} particle.
(10) Calculate the fitness value, $\phi(X_a(t_p+1))$, of the a^{th} particle.
(11) **if** $\phi(X_a(t_p+1)) < \phi(P_a)$ **then**
(12) $P_a \longleftarrow X_a(t_p+1)$
(13) **end if**
(14) **if** $\phi(X_a(t_p+1)) < \phi(P_g)$ **then**
(15) $P_g \longleftarrow X_a(t_p+1)$
(16) **end if**
(17) **end for**
(18) $t_p \longleftarrow t_p + 1$
(19) **end while**

Algorithm 1: Pseudocode of the PSO.

metric called distance value, $\rho(\overline{t}_i^a, \underline{t}_i)$, which combines the normalized constrained objective function (normalized delay) with the constraint violations. In the absence of feasible particles in the swarm, the distance value allows to distinguish the particles with low constraint violations from the ones with high constraint violations. By doing this, the search will progressively convey to the feasible region. The distance value is blended with a penalty term, $\beta(\overline{t}_i^a, \underline{t}_i)$, to form the unconstrained objective function, $\phi(\overline{t}_i^a, \underline{t}_i)$, and the candidate solutions are ranked based on their fitness values. Therefore, the unconstrained optimization formulation of the intersection motion coordination problem is defined as follows:

$$\text{minimize } \phi(\overline{t}_i^a, \underline{t}_i) = \rho(\overline{t}_i^a, \underline{t}_i) + \beta(\overline{t}_i^a, \underline{t}_i). \tag{16}$$

Starting with the formulation of the distance value, $\rho(\overline{t}_i^a, \underline{t}_i)$, the value of the objective function of the constrained problem—that is, the delay to the CR, $f(\overline{t}_i^a, \underline{t}_i)$—is normalized as follows:

$$\widehat{f}(\overline{t}_i^a, \underline{t}_i) = \frac{f(\overline{t}_i^a, \underline{t}_i) - \underline{f}}{\overline{f} - \underline{f}}, \tag{17}$$

where $\underline{f}$ and $\overline{f}$ are the minimum and maximum values of the constrained objective function amongst all the candidate solutions, respectively. Let N_f be the number of particles in the swarm that satisfy all the constraints of the optimization problem. The ratio of the particles that satisfy the constraints (feasible particles) is calculated by the following equation:

$$r_f = \frac{N_f}{N_p}. \tag{18}$$

The distance value, $\rho(\overline{t}_i, \underline{t}_i)$, is computed as follows:

$$\rho(\overline{t}_i^a, \underline{t}_i) = \begin{cases} \alpha(\overline{t}_i^a, \underline{t}_i), & \text{if } r_f = 0; \\ \sqrt{\widehat{f}(\overline{t}_i^a, \underline{t}_i) + \alpha(\overline{t}_i^a, \underline{t}_i)}, & \text{otherwise;} \end{cases} \tag{19}$$

where $\alpha(\overline{t}_i^a, \underline{t}_i)$ is the overall measure of all constraint violations, and it will be referred to as the *infeasibility measure* of a solution. A detailed description of the calculation of the infeasibility measure is given in Section 3.4.3. The self-adaptive penalty utilizes the population of feasible solutions in the swarm to control the penalty amount that will be added to the infeasible solutions. When the majority of the swarm is comprised of feasible particles, a penalty is added to the particles that have a high fitness value. In the case where the minority of the swarm's population consists of feasible solutions, the particles that have a large number of constraint violations are penalized significantly. Thus, the penalty term, $\beta(\overline{t}_i^a, \underline{t}_i)$, is defined as follows:

$$\beta(\overline{t}_i^a, \underline{t}_i) = \left(1 - r_f\right) \cdot \gamma(\overline{t}_i^a, \underline{t}_i) + r_f \cdot \zeta(\overline{t}_i^a, \underline{t}_i), \tag{20}$$

where

$$\gamma(\overline{t}_i^a, \underline{t}_i) = \begin{cases} 0, & \text{if } r_f = 0; \\ \alpha(\overline{t}_i^a, \underline{t}_i), & \text{otherwise;} \end{cases}$$
$$\zeta(\overline{t}_i^a, \underline{t}_i) = \begin{cases} 0, & \text{if } \overline{t}_i^a \text{ is a feasible particle;} \\ \widehat{f}(\overline{t}_i^a, \underline{t}_i), & \text{if } \overline{t}_i^a \text{ is an infeasible particle.} \end{cases} \tag{21}$$

If the infeasible solutions comprise of the swarm's majority, the first term, $\gamma(\overline{t}_i^a, \underline{t}_i)$, will dominate the penalty value. Therefore, each solution will be penalized based on its overall constraint violations. In the case of minor amount of infeasible solutions in the swarm, then the second term, $\zeta(\overline{t}_i^a, \underline{t}_i)$, will dominate the penalty value. The candidate

solutions that have a high fitness value will be added more penalty than the ones with a low value.

3.4. Severity of Constraint Violations: Calculating the Infeasibility Measure. The previous section described how the constrained optimization problem could be converted into an unconstrained one using the penalty function method. This conversion is deemed necessary because the PSO metaheuristic is applicable only to unconstrained optimization searches. The penalty function method relies on the incorporation of an auxiliary additive term, which is referred to as the *penalty term*, to the objective function. Its purpose is to retain the information collected by the infeasible solutions, instead of discarding it, such that they gradually converge to the feasible space.

The distance value $\rho(\overline{t}_i^a, \underline{t}_i)$ and the self-adaptive penalty term $\beta(\overline{t}_i^a, \underline{t}_i)$ of the unconstrained minimization problem, given in (16), require the determination of the infeasibility measure $\alpha(\overline{t}_i^a, \underline{t}_i)$. The latter should represent both the number of constraints that are activated and the severity to which each constraint is violated [62]. In this work, we adopt the self-adaptive penalty method presented in [55, 59, 62].

Contrary to a typical optimization search, the intersection control problem requires a sequence of intermediate steps before calculating the infeasibility measure of the candidate solutions. For a given candidate solution, the motion of the vehicle is projected to both the control and intersection regions to determine if the safety and kinematic constraints are violated.

A candidate arrival time constitutes a viable solution to the optimization problem if the incoming vehicle's trajectory satisfies specific requirements regarding its safety and motion characteristics. In summary, the safety requirements necessitate that two or more vehicles do not overlap (collide) while conveying in the control and intersection regions. The motion characteristic conditions limit the trajectory of the vehicles to certain acceptable velocity and acceleration profiles when the vehicle is transporting in the CR.

This section presents in detail the process that is used to define the infeasibility measure, of the self-adaptive penalty method, by quantifying the constraint violations of the candidate solutions. This process incorporates three intermediate procedures: the Trajectory Generation, the Motion Projection, and the Quantification of Constraint Violations. The subsequent steps are described for an arbitrary vehicle A_i that enters the IR at a time instant $\underline{t}_i$ and has a candidate arrival time $\overline{t}_i^a$.

3.4.1. Trajectory Generation. The first step towards the calculation of each solution's (vehicle's candidate arrival time) infeasibility measure is to generate the incoming vehicle's trajectory in the CR based on its candidate arrival time. Thus, each incoming vehicle's velocity profile, in the CR, is parameterized by its arrival time.

The trajectory of each vehicle in the CR is generated based on two kinematic conditions. The first requires that the vehicles shall not exceed their maximum velocity and acceleration bounds, $v_{\max}$ and $a_{\max}$, respectively. The second necessitates that the vehicles enter and exit the CR with their maximum velocity, $v_{\max}$. To minimize the number of configuration parameters of the trajectories, we limit the vehicle's velocity profile to have a trapezoidal form consisting of three distinct periods: constant deceleration, advancement with constant speed (zero acceleration), and constant acceleration (the deceleration period of the vehicle's trapezoidal velocity profile, within the CR, must proceed the acceleration period such that the vehicle's speed, at any instant, does not exceed its maximum admissible value, $v_{\max}$). Furthermore, all three periods have equal duration, and the absolute values of the deceleration and acceleration are also equal. Hence, the acceleration profile is comprised of two opposite pulses of same area.

Figure 8 illustrates the velocity and acceleration profiles of A_i and the sequence of the distinct acceleration periods. The expected time duration where A_i will be conveying in the CR is defined as follows:

$$t_g^a = \overline{t}_i^a - \underline{t}_i. \tag{22}$$

As mentioned earlier, the projected time duration of A_i in the CR, t_g^a, is further subdivided into three equal time intervals. Each interval corresponds to one of the vehicle's trajectory acceleration periods, and their respective duration is denoted by $t_q^a = t_g^a/3$. With the trapezoidal formulation of the velocity profile, the only parameter that needs to be determined is the vehicle's constant speed v_l^a during its zero acceleration phase (advancement with constant speed). This value is conveniently calculated by noting that the area under the velocity curve in the interval $[\underline{t}_i\ \overline{t}_i^a]$ is equal to the displacement of A_i during that time interval, i.e., the length of the CR, L. Using elementary calculations, the velocity of the vehicle at its zero acceleration period is given by

$$v_l^a = \frac{L - v_{\max} \cdot t_q^a}{2 \cdot t_q^a}. \tag{23}$$

The constant acceleration that is needed to bring the velocity of A_i from v_l^a to $v_{\max}$ in a linear manner, during the acceleration period, is calculated by the following equation:

$$\dot{v}_a = \frac{v_{\max} - v_l^a}{t_q^a}. \tag{24}$$

The above value also represents the slope of the velocity's straight line segment at the acceleration phase. The vehicle's constant deceleration, at the first phase of its trajectory, is equal to $-\dot{v}_a$. Finally, the velocity of A_i in the CR, for all three periods of the trajectory's profile, is expressed as follows:

$$v_i\left(t, \underline{t}_i, \overline{t}_i^a\right) = \begin{cases} v_{\max} - \dot{v}_a \cdot (t - \underline{t}_i), & \underline{t}_i \le t \le \underline{t}_i + t_q^a; \\ v_l^a, & \underline{t}_i + t_q^a < t \le \underline{t}_i + 2 \cdot t_q^a; \\ v_l^a + \dot{v}_a \cdot \left(t - \left(\underline{t}_i + 2 \cdot t_q^a\right)\right), & \underline{t}_i + 2 \cdot t_q^a < t \le \underline{t}_i + 3 \cdot t_q^a; \end{cases} \tag{25}$$

where $t \in [\underline{t}_i, \overline{t}_i^a]$.

3.4.2. Motion Projection. The second step involves the projection of the motion of each vehicle, in the control and

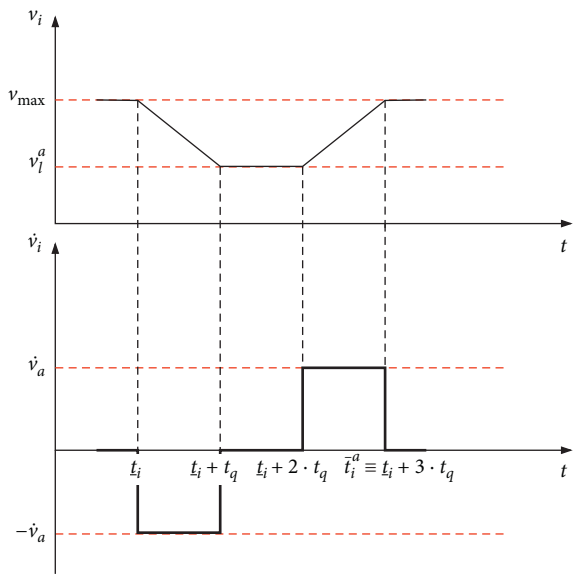

FIGURE 8: Velocity and acceleration profiles of vehicle A_i. The motion of the vehicles is broken down to three distinct periods: constant deceleration, advancement with constant speed, and constant acceleration.

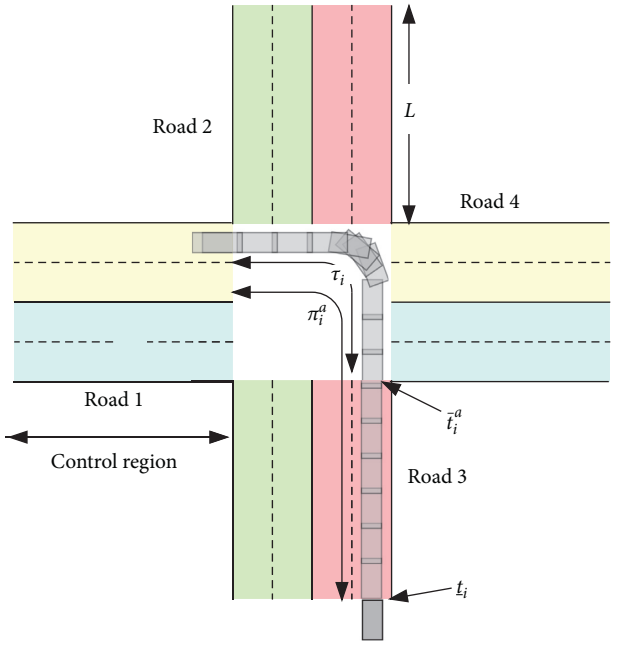

FIGURE 9: Illustration of the projected motion of vehicle A_i during its movement, with duration π_i^a, across the control and intersection regions, respectively, for the time instances of the set $\mathcal{T}_i^a$.

intersection regions, according to its generated velocity profile. It should be reminded that the vehicles transport through the IR with constant velocity. The projected motion, which corresponds to a candidate arrival time of the vehicle, is utilized to evaluate the part of the feasibility measure that pertains to its safety requirements.

To begin with, we calculate the total duration that each vehicle spends at the autonomous intersection. This duration is the sum of the time spans that the vehicle conveys in the control and intersection regions, respectively. Clearly, the vehicle's total duration at the intersection depends on its candidate arrival time. The total duration at the intersection π_i^a, of vehicle A_i and corresponds to the candidate arrival time $\bar{t}_i^a$, is calculated by the following equation:

$$\pi_i^a = \bar{t}_i^a - \underline{t}_i + \tau_i. \tag{26}$$

Using the kinematic expressions given in (2) and the generated velocity profile of (25), this procedure projects the motion of the vehicle in the time horizon π_i^a. Figure 9 illustrates how the Motion Projection procedure maps a vehicle's motion in the control and intersection regions. After determining the total duration π_i^a, a structure with all the discrete execution instances in the interval $[\underline{t}_i, \underline{t}_i + \pi_i^a]$ is formed; let $\mathcal{T}_i^a$ be the set of all these instances. The Motion Projection process finalizes by determining the set of postures, denoted by χ_i^a, that A_i will have at all instances of $\mathcal{T}_i^a$.

3.4.3. Quantification of Constraint Violations. The final step of the constraint handling process is dedicated to quantifying the level of violation of each constraint by a candidate solution. The objective is to capture the safety conflicts, in both the control and intersection regions, and also record violations of the vehicles' kinematic limitations. The identification of safety conflicts in the IR is performed similarly to the reservation paradigm that was introduced in [7].

The first constraint captures conflicts between vehicles in the IR that have not been identified at the preliminary search, e.g., the case where a vehicle overlays multiple cells. To capture these conflicts, initially, the resolution of the grid array is refined by a factor of $\mu \in \mathbb{Z}^+$. The refined grid array has $c_r = \mu \cdot (N_R/2) \cdot N_L$ columns, $r_r = \mu \cdot (N_R/2) \cdot N_L$ rows, and the cells' length is $e_r = e/\mu$. Each cell is identified by its row and column indexes as $E_{K,L}$, where $K \in \mathcal{R}_r = \{1, \ldots, r_r\}$ and $L \in \mathcal{C}_r = \{1, \ldots, c_r\}$. A timetable entry $B_{K,L}^r(t)$ is assigned to the refined cell $E_{K,L}$. If the latter is occupied at the execution instant t, then we set $B_{K,L}^r(t) = 1$; otherwise, $B_{K,L}^r = 0$. Before mapping the projected motion of A_i to the refined cells of the intersection, a space-buffer is applied to add a safety distance between the vehicles in the IR. The dimensions of the vehicle are artificially expanded by a factor $\kappa > 1$ such that its new dimensions become $l' = l \cdot \kappa$ and $w' = w \cdot \kappa$. The refined grid array of the IR is shown in Figure 10.

The occupancy mapping process, for A_i, entails the determination of the set of occupied cells $\mathcal{P}_i(t)$ in the refined IR that are encapsulated by the expanded dimensions of A_i at time t. Let $\mathcal{I}_i(t) = \{I_1, I_2, \ldots, I_{|\mathcal{P}_i|}\}$ be the set of indices that correspond to the elements (cells) of $\mathcal{P}_i(t)$. The number of violations that the candidate solution $\bar{t}_i^a$ attains, when A_i moves in the IR, is calculated by the following equation:

$$\eta_1(\bar{t}_i^a, \underline{t}_i) = \sum_{m \in \tau_n} \sum_{n=1}^{|\mathcal{I}_i(m)|} B_{I_n}^r(m), \tag{27}$$

where τ_n denotes the collection of all discrete execution instances that belong to the closed interval $[\bar{t}_i^a, \bar{t}_i^a + \tau_i]$. In the

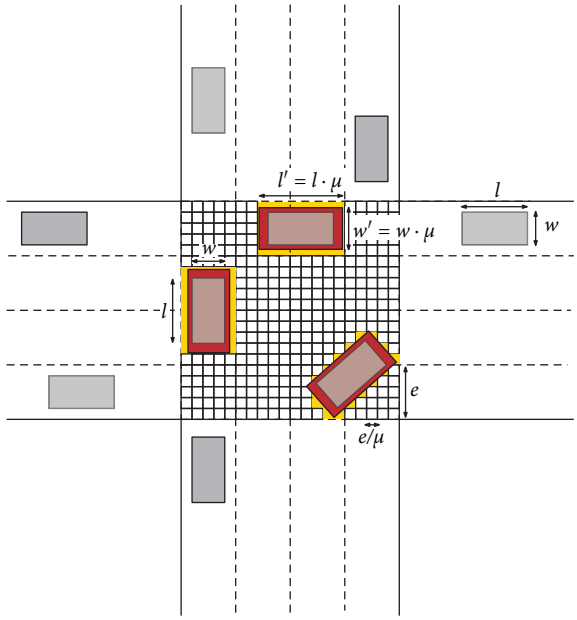

FIGURE 10: Illustration of the refined grid array at the IR. Inside the IR, the dimensions of the vehicles are artificially expanded as a means to introduce a space-buffer for their collision-free motion. The colored elements represent the occupied cells of the IR.

CR, the safety conflict between two same-lane vehicles is detected by inspecting the distance between them. The number of violations of A_i in the CR is given by

$$\eta_2(\overline{t}_i^a, \underline{t}_i) = \sum_{m \in \tau_c} b_c(m), \tag{28}$$

where τ_c denotes the set of execution instances in the closed interval $[\underline{t}_i, \overline{t}_i^a]$, which corresponds to the duration that A_i conveys in the CR, and b_c is a binary decision variable that signals whether or not there is a safety conflict between A_i and another vehicle in the lane. The decision variable b_c is defined as follows:

$$b_c(m) = \begin{cases} 0, & \text{if } l' - \left\| \mathcal{X}_i^a(m) - \mathcal{X}_p(m) \right\| \leq 0; \\ 1, & \text{otherwise.} \end{cases} \tag{29}$$

The term $\mathcal{X}_p$ represents the coordinates of the same lane, succeeding vehicles to A_i that convey in the CR at the time instant m, and l' (see definition of η_1) is the artificially expanded vehicles' length that adds a safe distance between vehicles in the CR. The remaining violations are assessed based on the kinematic constraints of the vehicles. Firstly, the (binary) acceleration violation value of the trapezoidal velocity's profile is given by

$$\eta_3(\overline{t}_i^a, \underline{t}_i) = \begin{cases} 0, & \text{if } \dot{v}_a \in [-a_{\max}, a_{\max}]; \\ 1, & \text{otherwise.} \end{cases} \tag{30}$$

In a similar fashion, the (binary) velocity violation value of the vehicle's trajectory is expressed as follows:

$$\eta_4(\overline{t}_i^a, \underline{t}_i) = \begin{cases} 0, & \text{if } v_l^a \in [0, v_{\max}]; \\ 1, & \text{otherwise.} \end{cases} \tag{31}$$

The overarching objective of the constraint handling method is to quantify all the constraint violations for each candidate solution. Therefore, the net value of the constraint violations of a candidate solution $\overline{t}_i^a$ is expressed as the weighted mean of all four constraint violation functions, given by

$$\alpha(\overline{t}_i^a, \underline{t}_i) = \frac{1}{4} \sum_{m=1}^{4} \frac{\eta_m(\overline{t}_i^a, \underline{t}_i)}{\eta_m^{\max}}, \tag{32}$$

where the term $\eta_m^{\max}$ is the maximum possible number of violations of the function η_m. The values of $\eta_1^{\max}$ of $\eta_2^{\max}$ are $|\tau_n| \cdot |\mathcal{I}_i(m)|$ and $|\tau_c|$, respectively. These two functions obtain their maximum value when the vehicle constantly overlaps with another one, when moving in the control and intersection regions, respectively. For the kinematic constraints, we have $\eta_3^{\max} = \eta_4^{\max} = 1$. A synoptic illustration in block diagrammatic form of the search process for the determination of the arrival time $\overline{t}_i$ of vehicle A_i, based on its entry time $\underline{t}_i$, is shown in Figure 11.

4. Computational Complexity of the IC

This section provides a detailed account of the computational complexity (or worst-case running time (we use the term computational complexity interchangeably with worst-case running time or time complexity. Formally, these terms are not equivalent [63]. The computational complexity of an algorithm refers to all resources required for executing it, which include both the size of memory (space complexity) and the time (time complexity) needed for running the algorithm. When the available resources are not specified, because the contrary would yield the analysis too complicated and not particularly insightful, the asymptotic running time of the algorithm is used to express its computational complexity)) of the IC with respect to the configuration parameters of the intersection, the geometry and motion characteristics of the vehicles, and the execution period of the algorithm (or else, its search resolution). More precisely, we delimit the time complexity needed by the IC to calculate the arrival time to the IR, $\overline{t}_i$, of an arbitrary vehicle A_i. This process is partitioned into two parts: the preliminary feasibility search, described in Section 3.1, and the search for an optimal arrival time, described in Sections 3.2–3.4. For convenience, let $\mathscr{C}_1$ be the computational complexity of the preliminary feasibility search and $\mathscr{C}_2$ the complexity of the optimization search. The total complexity, $\mathscr{C}$, of the procedure for calculating a vehicle's arrival time is $\mathscr{C} = \mathscr{C}_1 + \mathscr{C}_2$. In the subsequent analysis, we use standard properties of the big $\mathcal{O}$ notation and detail only the eventual complexity of the individual procedures involved in the calculation of the optimal arrival time. We intentionally omit the intermediate steps and the overjustification for the sake of brevity. The properties of the big $\mathcal{O}$ notation can be

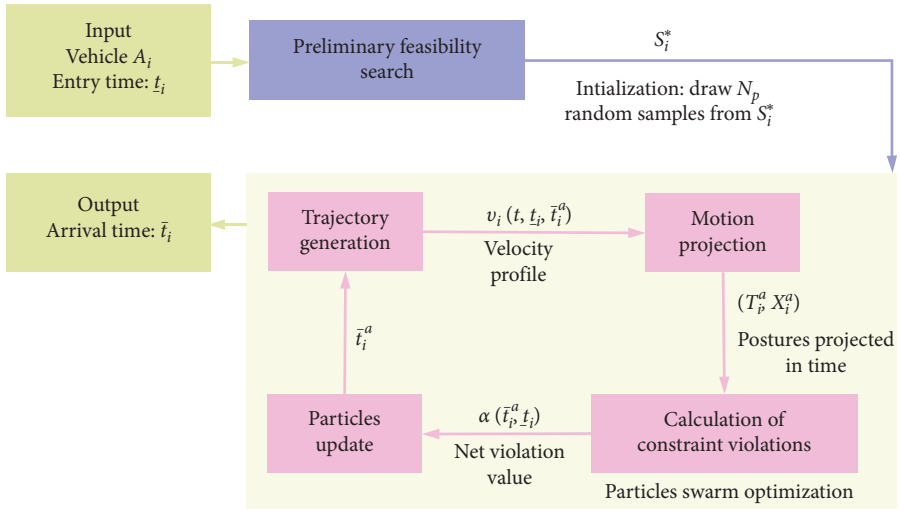

FIGURE 11: Block diagram of the procedures, and their respective interconnections, associated with the determination of the arrival time $\bar{t}_i$ for vehicle A_i based on the entry time $\underline{t}_i$.

found in standard textbooks on the theory of computation and discrete mathematics such as [64] and [65].

4.1. Complexity of the Preliminary Feasibility Search. The preliminary search for feasible solutions first identifies the fixed domain $[\bar{t}^e\ \bar{t}^l]$ of candidate arrival times to the IR; its duration is calculated by subtracting the value of the earliest arrival time, $\bar{t}^e$, given in (11), from the latest arrival time, $\bar{t}^l$, given in (9). Subsequently, for every discrete execution instant within this interval, we perform the feasibility test, expressed in (12), that singles out feasible arrival times based on the timetable of the intersection. The terms of the double summation in (12) are equal to the number of execution instances at which A_i transports within the IR. We have already established that the duration of the longest path in the IR is $\delta_{\max}$, described in (10). The longest possible path is traveled by the vehicles that transport at the outermost right lane of the CR and make a left turn at the IR. To this end, the maximum possible number of execution instances in the IR is $\delta_{\max}/T_s$, where T_s is the sampling interval of the IC. Hence, the complexity of the preliminary feasibility search is given by

$$\mathscr{C}_1 = \mathcal{O}\left(\frac{\bar{t}^l - \bar{t}^e}{T_s} \cdot \frac{\delta_{\max}}{T_s}\right) = \mathcal{O}\left(\left(\frac{e+l}{T_s v_{\max}}\right)^2 \left\lceil \frac{L}{l} \right\rceil (N_R N_L)^3\right). \tag{33}$$

From the above expression, we observe that the complexity of the preliminary feasibility search is cubic with respect to the product $N_R \cdot N_L$ (total number of lanes), which in part encapsulates the layout of the intersection. The complexity of the procedure is linear with respect to the floor term ($\lceil L/l \rceil$). The floor term ($\lceil L/l \rceil$) captures the impact of the control region's capacity to the complexity of the procedure. The higher the ratio (L/l) is, the more vehicles may simultaneously utilize the intersection, and the larger the search-space, $[\bar{t}^e\ \bar{t}^l]$, becomes. Finally, the term $(e+l)/T_s v_{\max}$ expresses the impact to the computational complexity of the relation between the vehicles motion characteristics (maximum velocity $v_{\max}$), the size of the intersection (width of lanes w), and the size of the vehicles (length l). Slow moving vehicles that transport over a sizable intersection result in more discrete execution instances that need to be processed for determining their feasibility as candidate solutions.

4.2. Complexity of the Optimization Process. The preliminary feasibility search procedure is followed by the execution of the PSO that forages the search-space seeking an optimal solution to the motion coordination problem. The complexity, $\mathscr{C}_2$, of the PSO is $\mathcal{O}(t_p^{\max} \cdot N_p \cdot \mathscr{C}_o)$, where $\mathscr{C}_0$ is the complexity required to compute the fitness value of the unconstrained objective function, $\phi(\cdot)$. The calculation of the objective function's fitness value, for a candidate solution, is outlined in Algorithm 2. The reader is reminded that $t_p^{\max}$ is the number of iterations and N_p is the number of particles in the swarm. The product term $t_p^{\max} \cdot N_p$ in $\mathscr{C}_2$ appears because of the iterative double for-loop responsible for the motion of the particles (see the pseudocode of the PSO algorithm given in Algorithm 1). We note that if a solution is not discovered throughout the prespecified number of iterations, $t_p^{\max}$ is incremented until the swarm finds one that satisfies the constraints of the problem.

The time complexity, $\mathscr{C}_0$—associated with the calculation of the unconstrained objective function's fitness value, $\phi(\bar{t}_i^a, \underline{t}_i)$, for a candidate solution $\bar{t}_i^a$—grows with the maximal duration of the time horizon for which the motion of the vehicle is projected at the intersection. The maximum span of this time horizon is $\tau_h = \bar{t}_i^l - \underline{t}_i + \delta_{\max}$, which is the sum of the longest possible duration needed for the vehicles to exit the CR plus the duration it takes them to convey the longest path in the IR. Within this time interval, the procedure that calculates the fitness value has to process τ_h/T_s discrete execution time instances.

The calculation of the objective function's fitness value is partitioned to three consecutive executable procedures (see Section 3.4). The Trajectory Generation is a static map;

Procedure: unconstrained objective function
Inputs: $\underline{t}_i, \overline{t}_i^a, L, v_{\max}, \mathcal{O}_i, B, B^r, \eta_m^{\max}, r_f, \overline{f}, \underline{f}$
Outputs: $\phi(\overline{t}_i^a, \underline{t}_i)$
(1) $[v_a, \dot{v}_a]$ = Trajectory Generation $[\underline{t}_i, \overline{t}_i^a, L, v_{\max}]$
(2) $[\chi_i^a]$ = Motion Projection $[v_a, \underline{t}_i, \overline{t}_i^a]$
(3) $[\eta_1, \eta_2, \eta_3, \eta_4]$ = Analysis of Constraint Violations $[B, B^r, x_i^a, y_i^a, v_a, \dot{v}_a]$
(4) $\alpha(\overline{t}_i^a, \underline{t}_i) \longleftarrow (1/4)\sum_{m=1}^{4} \eta_m(\overline{t}_i^a, \underline{t}_i)/\eta_m^{\max}$
(5) $\underline{f}(\overline{t}_i^a, \underline{t}_i) \longleftarrow \overline{t}_i^a - \underline{t}_i - t_s$
(6) $\widehat{f}(\overline{t}_i^a, \underline{t}_i) \longleftarrow (f(\overline{t}_i^a, \underline{t}_i) - \underline{f})/(\overline{f} - \underline{f})$
(7) **if** $r_f == 0$ **then**
(8) $\quad \rho(\overline{t}_i^a, \underline{t}_i) \longleftarrow \alpha(\overline{t}_i^a, \underline{t}_i)$
(9) $\quad \gamma(\overline{t}_i^a, \underline{t}_i) \longleftarrow 0$
(10) **else**
(11) $\quad \rho(\overline{t}_i^a, \underline{t}_i) \longleftarrow \sqrt{\widehat{f}(\overline{t}_i^a, \underline{t}_i) + \alpha(\overline{t}_i^a, \underline{t}_i)}$
(12) $\quad \gamma(\overline{t}_i^a, \underline{t}_i) \longleftarrow \alpha(\overline{t}_i^a, \underline{t}_i)$
(13) **end if**
(14) **if** $\alpha(\overline{t}_i^a, \underline{t}_i) > 0$ **then**
(15) $\quad \zeta(\overline{t}_i^a, \underline{t}_i) \longleftarrow \widehat{f}(\overline{t}_i^a, \underline{t}_i)$
(16) **else**
(17) $\quad \zeta(\overline{t}_i^a, \underline{t}_i) \longleftarrow 0$
(18) **end if**
(19) $\beta(\overline{t}_i^a, \underline{t}_i) \longleftarrow (1 - r_f) \cdot \gamma(\overline{t}_i^a, \underline{t}_i) + r_f \cdot \zeta(\overline{t}_i^a, \underline{t}_i)$
(20) $\phi(\overline{t}_i^a, \underline{t}_i) \longleftarrow \rho(\overline{t}_i^a, \underline{t}_i) + \beta(\overline{t}_i^a, \underline{t}_i)$

Algorithm 2: Pseudocode of the unconstrained objective function procedure. The procedure computes the fitness value of the objective function that corresponds to a candidate solution of the unconstrained optimization problem.

hence, its procedure does not contribute to the asymptotic time complexity of the algorithm. The Motion Projection calculates the posture of A_i for all execution instances in the time horizon τ_h; thus, its complexity is given by

$$\mathcal{O}\left(\frac{\tau_h}{T_s}\right) = \mathcal{O}\left(\frac{e+l}{T_s v_{\max}}\left\lceil\frac{L}{l}\right\rceil (N_R N_L)^2\right). \tag{34}$$

The final procedure, for determining the fitness value $\phi(\overline{t}_i^a, \underline{t}_i)$, is the Quantification of Constraint Violations, which is responsible for calculating the infeasibility measure, $\alpha(\overline{t}_i^a, \underline{t}_i)$, of a candidate solution $\overline{t}_i^a$. The complexity of this step is dictated by the asymptotic upper bounds on the computations that take place by the quantifying constraint violation functions η_j, where $j \in \{1, 2, 3, 4\}$ (see Section 3.4.3).

The constraint violation function η_1 processes all the execution instances for which the vehicle conveys in the IR. We have already shown that the number of these instances is upper bounded by $\delta_{\max}/T_s$. For each of these instances, the violation function η_1 goes over all points of the refined intersection array that are enclosed within the expanded dimensions of the vehicle (length l' and width w'). It is reminded that the length of each square "pixel" of the refined grid is e/μ; hence, the complexity for calculating η_1 is given by

$$\mathcal{O}\left(\frac{\delta_{\max}}{T_s} \cdot \frac{\kappa w}{e/\mu} \cdot \frac{\kappa l}{e/\mu}\right) = \mathcal{O}\left(\frac{e+l}{T_s v_{\max}} (N_R N_L)\left(\frac{\mu^2 \kappa^2 w l}{e^2}\right)\right). \tag{35}$$

The second constraint violation function, η_2, takes place for all execution instances for which the vehicle moves within the CR. The maximum duration of this interval, for a vehicle A_i, is $\overline{t}_i^l - \underline{t}_i$. By virtue of (9), which expresses the value of the latest arrival time to the IR, the complexity of this step is given by

$$\mathcal{O}\left(\frac{\overline{t}_i^l - \underline{t}_i}{T_s}\right) = \mathcal{O}\left(\frac{e+l}{T_s v_{\max}}\left\lceil\frac{L}{l}\right\rceil (N_R N_L)^2\right). \tag{36}$$

Finally, the violations of the kinematic constraints (velocity and acceleration) are calculated from the binary functions η_3 and η_4, respectively. Because these functions are static—that is, their computation time does not vary—they have constant time complexity, $\mathcal{O}(1)$, and do not add asymptotic time complexity of the procedure.

Combining the complexities of the three intermediate procedures (we have used the rule-of-sums property for big $\mathcal{O}$; see [65]: if $g_1 \in \mathcal{O}(f_1)$ and $g_2 \in \mathcal{O}(f_2)$, then $g_1 + g_2 \in \mathcal{O}(\max\{f_1, f_2\})$) (namely, Trajectory Generation, Motion Projection, and Quantification of Constraint Violations), the complexity $\mathscr{C}_0$ associated with the calculation of a single fitness value of the objective function is given by

$$\mathscr{C}_0 = \mathcal{O}\left(\frac{e+l}{T_s v_{\max}} (N_R N_L)^2 \left\{\left\lceil\frac{L}{l}\right\rceil + \frac{wl}{e^2}(\mu\kappa)^2\right\}\right). \tag{37}$$

The above expression validates the anticipated expectation that the computational complexity increases—in a quadratic fashion—with the granularity of the collision search in the IR, encapsulated by the product $\mu \cdot \kappa$. To calculate the complexity of the entire optimization step, $\mathscr{C}_0$ is multiplied by $t_p^{\max} \cdot N_p$, rendering the complexity of the optimization step to (we have used the transitivity property

for big $\mathcal{O}$; see [64, 65]: if $h \in \mathcal{O}(g)$ and $g \in \mathcal{O}(f)$, then $h \in \mathcal{O}(f)$):

$$\mathscr{C}_2 = \mathcal{O}\left(t_p^{\max} N_p \frac{e+l}{T_s v_{\max}} (N_R N_L)^2 \left\{\left\lceil \frac{L}{l} \right\rceil + \frac{wl}{e^2}(\mu\kappa)^2\right\}\right). \tag{38}$$

In practice, due to the product $t_p^{\max} \cdot N_p$, the optimization process is significantly more computationally demanding compared to the preliminary feasibility search, despite that the complexity of the latter is cubic with respect to $N_R \cdot N_L$. The quadratic term of the ratio $e + l/T_s v_{\max}$ that appears in $\mathscr{C}_1$ does not have significant impact when the distance covered by the vehicles in a single step (distance $T_s \cdot v_{\max}$) is comparable to their dimensions (length l) and the size of the intersection's cells (width e).

4.3. Total Complexity: Single Vehicle. The time complexity, $\mathscr{C}$, of the comprehensive procedure that computes the arrival time of an incoming vehicle to the IR is given by

$$\mathscr{C} = \mathcal{O}\left(t_p^{\max} N_p \left(\frac{e+l}{T_s v_{\max}}\right)^2 (N_R N_L)^3 \left\{\left\lceil \frac{L}{l} \right\rceil + \frac{wl}{e^2}(\mu\kappa)^2\right\}\right). \tag{39}$$

The above asymptotic upper bound incorporates all the dominant configuration parameters that pertain to the intersection control problem, where a change to their value will impact the running time of the algorithm. The complexity of the algorithm depends on the physical dimensions and configuration of the intersection, encapsulated by the parameters N_R, N_L, and e; the geometry and motion characteristics of the vehicles, captured by the parameters l, w, and $v_{\max}$; and parameters of the algorithm, namely, $t_p^{\max}$, N_p, μ, and κ.

5. Simulation Results

In this section, we evaluate the performance and effectiveness of the proposed IC via numerical simulations. The simulations were carried out using the MATLAB computing environment. A comparison of the proposed IC with the existing state of the art, as it appears in the literature (e.g., [7, 26, 66]), was deemed impractical. In the existing literature, the intersection control problem is staged in various dissimilar ways, making the direct comparison between methodologies unfeasible. Some of its variants are limiting the problem to single-lane roads, vehicles that have their motion governed by linear kinematics, or are adopting a vehicle-to-vehicle (decentralized) instead of a vehicle-to-infrastructure (centralized) architecture. Thus, we compare the performance of the proposed IC with a conventional traffic lights system and validate its efficacy for varying intersection configurations and traffic loads.

The parameters of the numerical simulations are outlined in Table 1. The length, l, and width, w, of the autonomous vehicles are set to 6 m and 3 m, respectively. The maximum velocity of the vehicles, $v_{\max}$, was set to 10 m/s, and their accelerate/decelerate bound, to ± 2 m/s^2. The width of the lanes, e, was selected equal to the vehicles width, w, and the resolution factor for detecting collisions, μ, was set to 10 such that $e_r = 0.3$ m. Finally, the sampling interval of the IC, T_s, was selected as 0.01 s.

Table 1: Numerical simulation parameters.

Simulation parameters	
L	6 m
w	3 m
$v_{\max}$	10 m/s
$a_{\max}$	2 m/s^2
e	0.3 m
μ	10
T_s	0.01 s
c_1	2.05
c_2	2.05

The performance of the IC is analyzed for two different intersection configurations. For each configuration, we examine the delay of the vehicles in the CR for varying traffic loads in the roads of the intersection. The configuration of the first intersection involves two roads with a single-lane each—the vehicles can only move straight, and there are no turns. The two single-lane road intersection configuration ($N_R = 2$ and $N_L = 1$) has a constant service time $\tau_i = ((e+l)/v_{\max})$ s for every incoming vehicle. The entry times of the incoming vehicles, at each lane of the CR, are randomly generated by a Poisson process. The rate of the Poisson process (or arrival intensity), λ, is gradually increased to simulate different traffic loads at the intersection. Light, medium, and heavy traffic conditions were reproduced by implementing an arrival intensity, λ, of 10, 30, and 50 vehicles per minute (vpm), respectively.

Figure 12 depicts the trajectory of the vehicles while they are moving in the CR. At the low and medium traffic loads, vehicles traverse the CR with almost zero delay. However, the heavy traffic load causes the vehicles to experience a minor delay with an average value of approximately 9.8 s. Figure 13 compares the traffic flow of the vehicles at the entrance versus the exit of the CR, for different values of the arrival intensity, λ. The difference in the traffic flow reflects the impedance (delay) of the vehicles in the CR. The two traffic flows, at the entrance and exit of the CR, are nearly equivalent when λ is less and equal to 30 vpm. A reduction of roughly 10% in the exit flow, compared to the entry, appears at the maximum traffic load, when the arrival intensity is 50 vpm.

Figure 14 depicts the utilization of the IR by the incoming vehicles. For the two single-lane road intersection configuration ($N_R = 2$ and $N_L = 1$), the IR is comprised of a single cell. Its utilization is defined as the percentage of time that the IR is occupied by a vehicle over the system's entire duration of the operation. The overarching goal is not only to minimize the delay of the vehicles but also to schedule the intersection's resources efficiently for mitigating idle

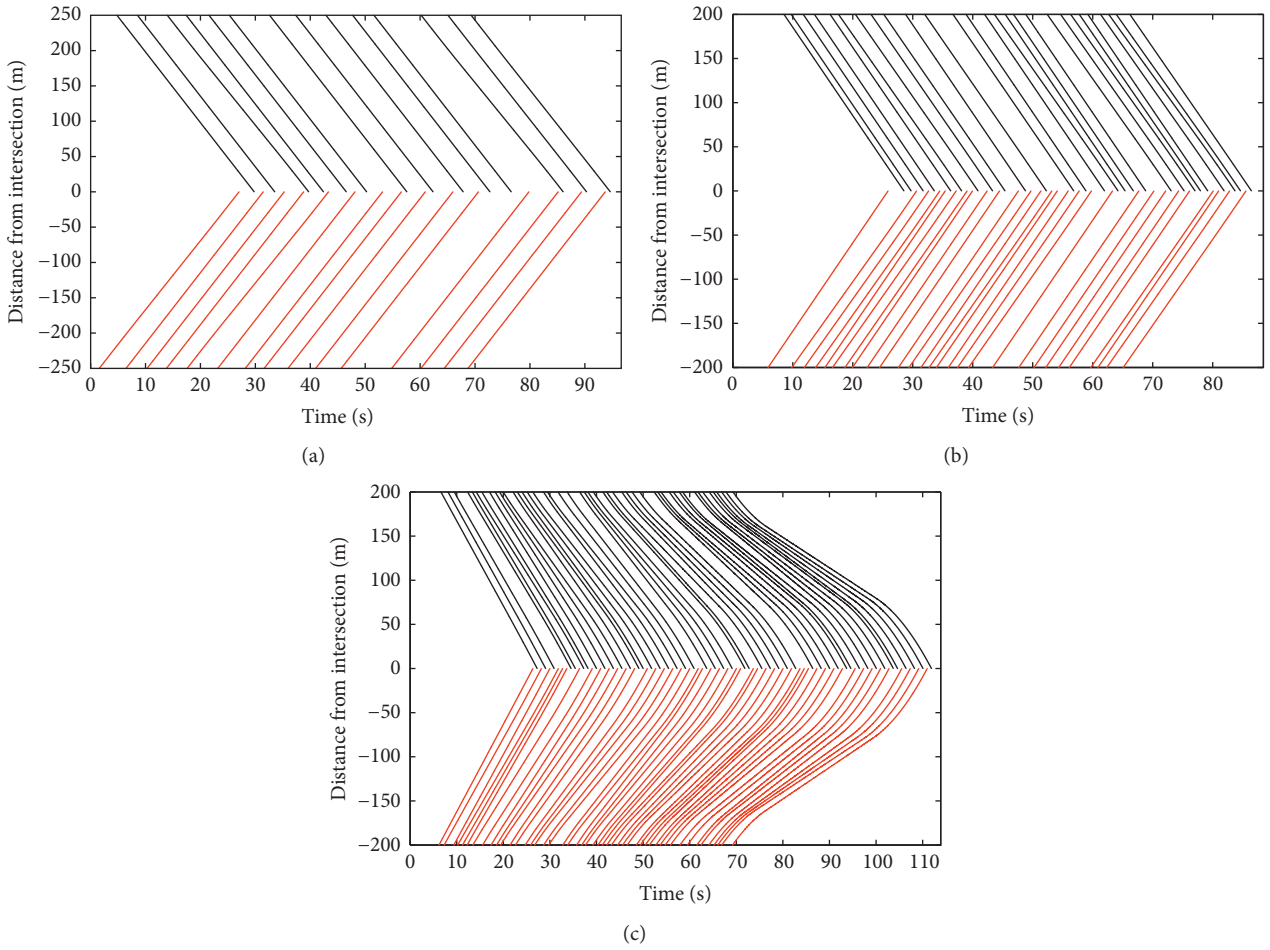

Figure 12: First case study: trajectories of the vehicles inside the CR with respect to time, for different traffic intensities. The top, middle, and bottom graphic plots represent the low, medium, and heavy traffic loads with arrival intensities 10, 30, and 50 vpm, respectively. The upper half of the plot corresponds to vehicles of Road 1, while the lower half to vehicles of Road 2.

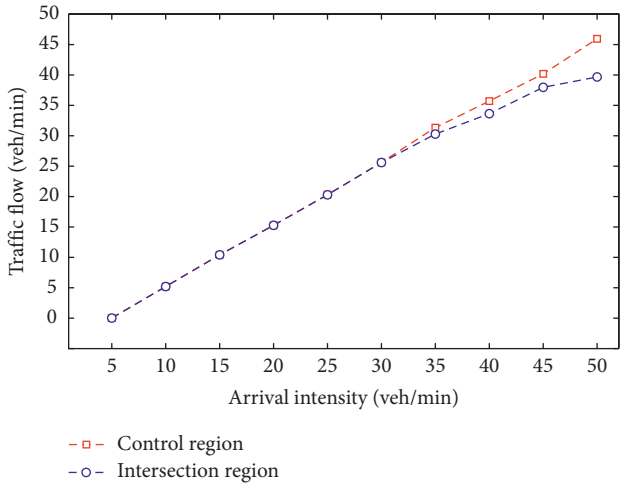

Figure 13: First case study: average traffic flows at the entry and exit of the CR, respectively, for varying values of the arrival intensity, λ.

time—that is, attain full utilization (100%) when the traffic is heavy. The results show that the IC controller accomplishes almost full utilization when the arrival intensity of the vehicles is greater and equal to 40 vpm.

The second simulation case study accounts an intersection configuration where four roads with two lanes each are intersecting ($N_R = 4$ and $N_L = 2$), and the vehicles can make a turn in either direction. In this configuration, the

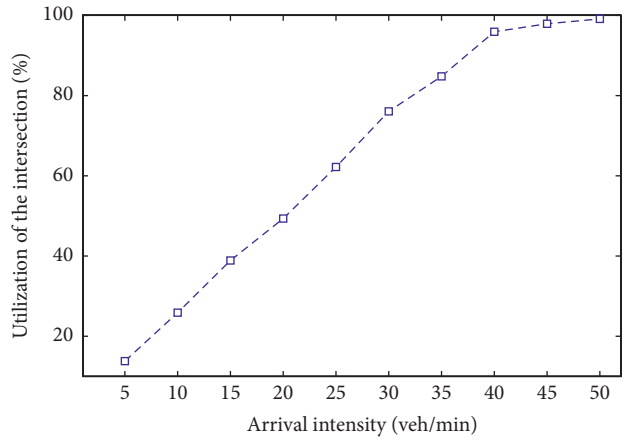

Figure 14: First case study: intersection's utilization for varying values of the arrival intensity, λ.

number of roads is doubled, compared to the first simulation case study, resulting in a substantial increase in the vehicles traveling time. The arrival intensity, in each lane, is set to 10, 20, and 30 vpm. For this case study, the intensity values are lower compared to the first scenario. However, in this configuration, the number of lanes has been quadrupled compared to the first case; thus, the congestion is considerable, especially when the arrival intensity is 30 vpm.

The route of each vehicle at the intersection was determined by two consecutive Bernoulli processes. Initially, the vehicle determines if a turn takes place with a probability $p_t = 0.3$. In the case of a turn, the selection of the exit road is also random. This probability depends on the lane of the vehicle. For a left lane vehicle, the probabilities for turning left and right are $p_f^l = 0.7$ and $p_f^r = 0.3$, respectively. For a right lane vehicle, the probabilities p_f^l and p_f^r are reversed.

The performance of the IC is compared with a conventional traffic lights intersection control system. The traffic lights signaling introduces stop-and-go waves, and it is implemented in three consecutive phases. In the first phase, the green light is on for only the vehicles that are on parallel roads and going to turn left. Then, the vehicles that are going straight are allowed to move, but the left turn is prohibited during this period. The final phase is the amber light, and it provides a safe transition between the red and green lights. The amber light enables vehicles to finalize their motion in the IR before the next nonintersecting roads are mobilized.

Figure 15 depicts the average delay that the vehicles experience along the CR. The IC allows the vehicles to traverse the CR with an average delay that is less than 10 s, until the traffic load takes its highest value. The delay of the vehicles in the intersection reaches its maximum value, of approximately 14 s, when the vehicle arrival rate is 30 vpm per lane. The simulation results show that the proposed IC outperforms the traffic lights control by reducing the average delay by approximately in half. Figure 16 depicts the average traffic flow at the entry and exit of the intersection CR. The incoming and outgoing traffic flows are almost the same (negligible delay), and their difference is at most 16% when $\lambda = 30$ vpm.

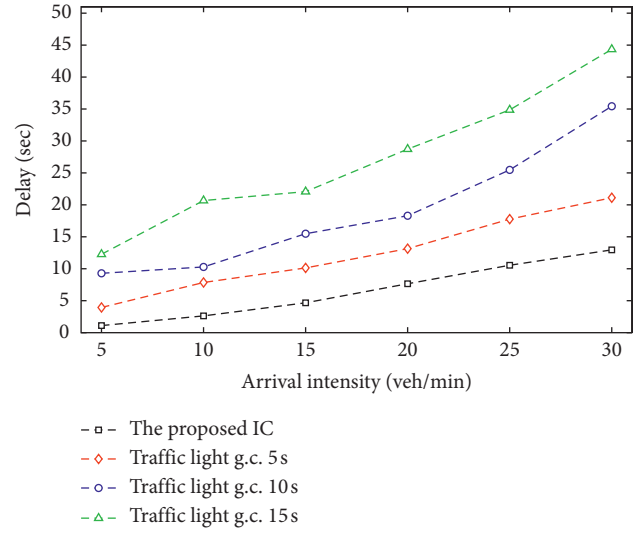

Figure 15: Second case study: delay at the traffic intersection under the proposed IC and conventional traffic signal control with green cycles (g.c.s) of 5 s, 10 s, and 15 s, respectively.

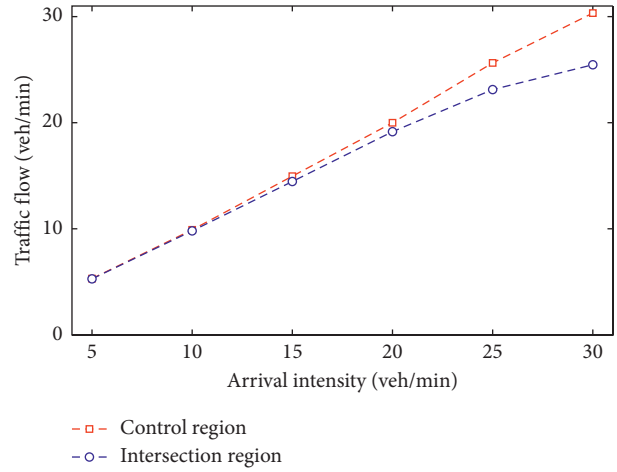

Figure 16: Second case study: average traffic flow at the entry and exit of the CR, respectively, for varying values of the arrival intensity, λ.

6. Conclusion

We have presented a scheduling-based approach for the formulation of an IC for autonomous vehicles that cross a multilane intersection. The multivehicle motion coordination task at the intersection is cast as an optimization problem with nonlinear constraints, where the optimization variable is the vehicle's arrival time at the CR. The IC computes the trajectory of each incoming vehicle in a first-come first-served order. The performance metric of the optimization problem is the delay of the vehicle at the CR. The penalty method is used to convert the constrained optimization problem into an unconstrained one, making it compatible with metaheuristics. The fitness value of the unconstrained problem's objective function captures the severity of the violations that correspond to a candidate solution. The PSO is employed to harvest the search-space for an optimal solution to the optimization problem.

The IC relies on model-based heuristic computation that merges effectively combinatorics, optimization, and dynamics equations into a concrete framework that makes analytically derived optimal policies redundant. We provide a detailed account of the computational complexity of the IC as a function of its configuration parameters and the intersection's layout. It is shown that the IC is cubic to the total number of lanes in the intersection and linear to the number of particles of the PSO. Simulation results demonstrate that the IC outperforms conventional traffic signals control, reducing significantly the average delay that the vehicles experience at the intersection, especially when the traffic load is medium to heavy.

Contrary to a conventional optimization task, the intersection control requires the forward projection of the vehicle's trajectory, which corresponds to a candidate solution. Because the PSO simulates multiple candidate solutions at each iteration, the IC is subjected to considerable computational strain. Future work involves modifications to the existing algorithm that will reduce the computational complexity of the IC, for example, a better selection of the initial candidate solutions that will accelerate the convergence of the PSO to an optimal solution.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

This article is the complete and detailed version of the work by the same authors reported in the Proceedings of 2018 IEEE Annual American Control Conference (ACC) [67].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Talebpour and H. S. Mahmassani, "Influence of connected and autonomous vehicles on traffic flow stability and throughput," *Transportation Research Part C: Emerging Technologies*, vol. 71, pp. 143–163, 2016.

[2] G. R. de Campos, P. Falcone, R. Hult, H. Wymeersch, and J. Sjöberg, "Traffic coordination at road intersections: autonomous decision-making algorithms using model-based heuristics," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 8–21, 2017.

[3] J. Rios-Torres and A. A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1066–1077, 2017.

[4] National Highway Traffic Safety Administration, "Motor vehicle crashes: overview," *Traffic Safety Facts Research Note*, vol. 2016, pp. 1–9, 2015.

[5] J. Broughton, P. Thomas, A. Kirk et al., *Traffic Safety Basic Facts 2012: Junctions*, 2013.

[6] Directive 2010/40/EU of the European parliament and of the council," *Official Journal of the European Union*, vol. 50, p. 207, 2010.

[7] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, 2008.

[8] X. Qian, F. Altché, J. Grégoire, and A. de La Fortelle, "Autonomous intersection management systems: criteria, implementation and evaluation," *IET Intelligent Transport Systems*, vol. 11, no. 3, pp. 182–189, 2017.

[9] M. Athans, "A unified approach to the vehicle-merging problem," *Transportation Research*, vol. 3, no. 1, pp. 123–133, 1968.

[10] W. Levine and M. Athans, "On the optimal error regulation of a string of moving vehicles," *IEEE Transactions on Automatic Control*, vol. 11, no. 3, pp. 355–361, 1966.

[11] S. A. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 7, pp. 1646–1651, 2010.

[12] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 145–154, ACM, Montreal, Canada, 2012.

[13] L. Makarem and D. Gillet, "Fluent coordination of autonomous vehicles at intersections," in *Proceedings IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2557–2562, Seoul, South Korea, October 2012.

[14] Y. J. Zhang, A. A. Malikopoulos, and C. G. Cassandras, "Optimal control and coordination of connected and automated vehicles at urban traffic intersections," in *Proceedings of the American Control Conference*, pp. 6227–6232, Boston, MA, USA, July 2016.

[15] L. Chen and C. Englund, "Cooperative intersection management: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2016.

[16] K. M. Dresner, *Autonomous Intersection Management*, Ph.D. thesis, University of Texas at Austin, Austin, TX, USA, 2009.

[17] M. A. S. Kamal, J.-i. Imura, T. Hayakawa, A. Ohata, and K. Aihara, "A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights," *IEEE*

Transactions on Intelligent Transportation Systems, vol. 16, no. 3, pp. 1136–1147, 2015.

[18] K.-D. Kim and P. R. Kumar, "An mpc-based approach to provable system-wide safety and liveness of autonomous ground traffic," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3341–3356, 2014.

[19] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *Proceedings of the American Control Conference*, pp. 763–768, Chicago, IL, USA, July 2015.

[20] F. Yan, M. Dridi, and A. El Moudni, "Autonomous vehicle sequencing algorithm at isolated intersections," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, pp. 1–6, St. Louis, MO, USA, October 2009.

[21] Q. Jin, G. Wu, K. Boriboonsomsin, and M. Barth, "Multi-agent intersection management for connected vehicles using an optimal scheduling approach," in *Proceedings of the IEEE International Conference on Connected Vehicles and Expo*, pp. 185–190, Beijing, China, December 2012.

[22] J. Wu, F. Perronnet, and A. Abbas-Turki, "Cooperative vehicle-actuator system: a sequence-based framework of cooperative intersections management," *IET Intelligent Transport Systems*, vol. 8, no. 4, pp. 352–360, 2014.

[23] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.

[24] J. Lee, B. Park, K. Malakorn, and J. So, "Sustainability assessments of cooperative vehicle intersection control at an urban corridor," *Transportation Research Part C: Emerging Technologies*, vol. 32, pp. 193–206, 2013.

[25] Q. Jin, G. Wu, K. Boriboonsomsin, and M. Barth, "Platoon-based multi-agent intersection management for connected vehicle," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pp. 1462–1467, The Hague, Netherlands, October 2013.

[26] D. Miculescu and S. Karaman, "Polling-systems-based autonomous vehicle coordination in traffic intersections with no traffic signals," 2016, http://arxiv.org/abs/1607.07896.

[27] J. Gregoire and E. Frazzoli, "Hybrid centralized/distributed autonomous intersection control: using a job scheduler as a planner and inheriting its efficiency guarantees," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 2549–2554, Las Vegas, NV, USA, December 2016.

[28] M. Van Middlesworth, K. Dresner, and P. Stone, "Replacing the stop sign: unmanaged intersection control for autonomous vehicles," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1413–1416, Estoril, Portugal, May 2008.

[29] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection management using vehicular networks," Tech. Rep., SAE Technical Paper, Detroit, MI, USA, 2012.

[30] J. Gregoire, S. Bonnabel, and A. de La Fortelle, *Optimal Cooperative Motion Planning for Vehicles at Intersections*, 2013.

[31] X. Qian, J. Gregoire, F. Moutarde, and A. De La Fortelle, "Priority-based coordination of autonomous and legacy vehicles at intersection," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pp. 1166–1171, Qingdao, China, October 2014.

[32] G. R. Campos, P. Falcone, H. Wymeersch, R. Hult, and J. Sjöberg, "Cooperative receding horizon conflict resolution at traffic intersections," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 2932–2937, Los Angeles, CA, USA, December 2014.

[33] S. Scellato, L. Fortuna, M. Frasca, J. Gómez-Gardeñes, and V. Latora, "Traffic optimization in transport networks based on local routing," *The European Physical Journal B*, vol. 73, no. 2, pp. 303–308, 2010.

[34] A. Buscarino, L. Fortuna, M. Frasca, and A. Rizzo, "Dynamical network interactions in distributed control of robots, Chaos," *An Interdisciplinary Journal of Nonlinear Science*, vol. 16, no. 1, Article ID 015116, 2006.

[35] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing, Boston, MA, USA, 1st edition, 1989.

[36] F. Altch and A. de La Fortelle, "Analysis of optimal solutions to robot coordination problems to improve autonomous intersection management policies," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 86–91, Gothenburg, Sweden, June 2016.

[37] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[38] A. Bicchi, G. Casalino, and C. Santilli, "Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles," in *Proceedings of the International Conference on Robotics and Automation*, vol. 2, pp. 1349–1354, Nagoya, Japan, May 1995.

[39] J. Kennedy and R. Eberhart, "Particle swarm optimization (PSO)," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Western Australia, Perth, 1995.

[40] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the IEEE Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, October 1995.

[41] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[42] K.-L. Du and M. Swamy, *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*, Springer, Berlin, Geramny, 2016.

[43] Q. He, L. Wang, and B. Liu, "Parameter estimation for chaotic systems by particle swarm optimization," *Chaos, Solitons & Fractals*, vol. 34, no. 2, pp. 654–661, 2007.

[44] Z. Bingül and O. Karahan, "A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control," *Expert Systems with Applications*, vol. 38, no. 1, pp. 1017–1031, 2011.

[45] A. Chatterjee, K. Pulasinghe, K. Watanabe, and K. Izumi, "A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 6, pp. 1478–1489, 2005.

[46] M. P. Wachowiak, R. Smolikova, Y. Zheng, J. M. Zurada, and A. S. Elmaghraby, "An approach to multimodal biomedical image registration utilizing particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 289–301, 2004.

[47] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen, "Feature selection based on rough sets and particle swarm optimization," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 459–471, 2007.

[48] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 3, pp. 771–779, 2004.

[49] Y. Yuan, Z. He, and M. Chen, "Virtual MIMO-based cross-layer design for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 3, pp. 856–864, 2006.
[50] G. K. Venayagamoorthy and W. Zha, "Comparison of non-uniform optimal quantizer designs for speech coding with adaptive critics and particle swarm," *IEEE Transactions on Industry Applications*, vol. 43, no. 1, pp. 238–244, 2007.
[51] B. Liu, L. Wang, and Y.-H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 18–27, 2007.
[52] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, Article ID 685175, 10 pages, 2008.
[53] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 69–73, Xi'an, China, August 1998.
[54] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 101–106, Seoul, South Korea, 2001.
[55] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 561–579, 2010.
[56] S. Sivanandam and S. Deepa, *Introduction to Genetic Algorithms*, Springer Science & Business Media, Berlin, Germany, 2007.
[57] S. S. Rao, *Engineering Optimization: Theory and Practice*, John Wiley & Sons, Hoboken, NJ, USA, 2009.
[58] Ö. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, 2005.
[59] B. Tessema and G. G. Yen, "A self adaptive penalty function based algorithm for constrained optimization," in *Proceedings IEEE Congress on Evolutionary Computation*, pp. 246–253, Vancouver, Canada, July 2006.
[60] M. Schoenauer and S. Xanthakis, "Constrained ga optimization," in *ICGA*, pp. 573–580, Morgan Kauffman Publishers, Burlington, MA, USA, 1993.
[61] O. B. Haddad and M. A. Mariño, "Dynamic penalty function as a strategy in solving water resources combinatorial optimization problems with honey-bee mating optimization (hbmo) algorithm," *Journal of Hydroinformatics*, vol. 9, no. 3, pp. 233–250, 2007.
[62] R. Farmani and J. A. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 445–455, 2003.
[63] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, USA, 2009.
[64] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley Longman Publishing, Boston, MA, USA, 2nd edition, 1994.
[65] M. J. Dinneen, G. Gimel'farb, and M. C. Wilson, *Introduction to Algorithms, Data Structures & Formal Languages*, Creative Commons, Mountain View, CA, USA, 2016.
[66] A. A. Malikopoulos, C. G. Cassandras, and Y. J. Zhang, "A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections," *Automatica*, vol. 93, pp. 244–256, 2018.
[67] M. A. Guney and I. A. Raptis, "Scheduling-driven motion coordination of autonomous vehicles at a multi-lane traffic intersection," in *Proceedings of the 2018 Annual American Control Conference (ACC)*, pp. 4038–4043, Milwaukee, WI, USA, June 2018.