Perspectives in Mathematics and Applications
e-ISSN 3110-3847 | Vol. 02 No. 01 (2026), Pages 37-55
DOI https://doi.org/10.66256/permata.v2i1.44

A unified graph-based optimization model combining shortest paths and Hamiltonian constraints for motorsport strategy

Hanif Intishar Razan[1], Humaira Herwidya Nisa[2], Muhammad Rizki Eka Darmawan[2]
[1] Universitas Presiden, Bekasi, Indonesia.
[2] Mathematics Departments, Universitas Islam Negeri Siber Syekh Nurjati Cirebon, Indonesia

Article Info

Article history:

Received 01 21, 2026
Accepted 05 31, 2026
Published 06 20, 2026

Keywords:

Graph-based Optimization
Motorsport Strategy
Shortest Path
Hamiltonian Constraints
Racing Line Optimization

ABSTRACT

This study proposes a unified graph-based optimization model that combines shortest-path methods with Hamiltonian constraints to optimize motorsport strategy. Existing approaches address racing line optimization and pit stop decision-making as separate problems, leaving the formal integration of trajectory-level and strategy-level optimization within a single graph-theoretic architecture as an open problem. The proposed framework models racing trajectories using layered weighted graphs with edge costs derived from track curvature and vehicle dynamics, while strategic decisions are represented within a constrained state-space graph ensuring non-redundant coverage of critical race phases. Shortest-path optimization is employed to minimize the total race time under physically informed constraints. Numerical simulations show that A* reduces lap time by 0.07 seconds over Dijkstra (87.29 s vs. 87.36 s), and the two-stop pit strategy yields the shortest total race time (832.85 s), outperforming the one-stop (896.5 s) and no-stop (879.75 s) alternatives. However, the model is limited by a linear assumption of tire degradation and deterministic race conditions, which may not fully capture stochastic on-track interactions. The originality of this work lies in the formal coupling of trajectory-level edge weights with Hamiltonian-constrained strategy search within a single optimization architecture — a unification not previously established in the motorsport optimization literature. These findings demonstrate graph-based mathematical modeling as a tractable and extensible foundation for data-driven motorsport strategy.

Corresponding Author:

Humaira Herwidya Nisa
Mathematics Departments, Universitas Islam Negeri Siber Syekh Nurjati Cirebon, Indonesia
email: humairaamaira767@gmail.com

1. INTRODUCTION

Motorsport strategy optimization is a challenging decision-making problem where small performance improvements can make a big difference in deciding competition winners. In Formula 1 (F1) racing, a race's outcome is affected not only by the track's geometry but also by strategic choices such as tire selection, pit-stop timing, energy management, and traffic management. These decisions are subject to strict physical constraints, regulatory limits, and time dependencies, making analytical optimization difficult. Prior to the 2025 season, new rules regarding aerodynamics and the use of hybrid power units are applying more pressure on teams to perfect their race lines and pit stop strategies, especially for teams such as Scuderia Ferrari, who are battling for the constructors' title. As race strategies become increasingly data-driven, there is an increasing need for mathematically rigorous frameworks that integrate trajectory optimization and strategic planning in a unified model.

Graph-theoretic methods have been independently applied to both sub-problems. Shortest path algorithms on weighted graphs have been used to optimize racing trajectories, with edge costs derived from curvature and vehicle dynamics [1]. Separately, discrete state-space graphs have been used to model pit stop sequencing and energy allocation across race phases [2]. Despite their effectiveness, such approaches are typically treated as separate optimization problems, yielding fragmented solutions that fail to capture the interdependence between trajectory-level and strategy-level decisions.

This fragmentation represents the central gap that the present study aims to address. Although classical graph concepts such as shortest-path algorithms and Hamiltonian-path constraints have each been studied extensively in discrete mathematics and algorithmic optimization [3, 4], their combined application to motorsport strategy remains largely unexplored. In particular, it has not been established how Hamiltonian structural constraints can be embedded in a shortest-path formulation to enforce non-redundant traversal of critical race phases without increasing computational complexity to an intractable degree.

Two research questions follow from this gap. First, can a unified graph model that couples a layered geometric trajectory graph with a constrained strategic state-space graph produce strategy decisions that are jointly optimal at both the trajectory and race-phase levels? Second, do Hamiltonian constraints on strategic decision nodes improve the stability and coherence of pit stop sequencing under tire degradation conditions compared to unconstrained shortest path search?

This paper addresses both questions by proposing a unified graph-based optimization framework that integrates racing line optimization and race strategy planning within a single mathematical architecture. By embedding Hamiltonian constraints directly into the strategic graph's feasibility structure, the model ensures that each critical race phase is visited exactly once, a property neither guaranteed nor targeted by prior shortest-path approaches. The contribution of this work differs from existing studies [1, 2, 5, 6] not in the use of Dijkstra or A* per se, but in the formal coupling of trajectory-level edge weights with Hamiltonian-constrained strategy search, which enables coherent optimization across spatial and temporal decision layers simultaneously.

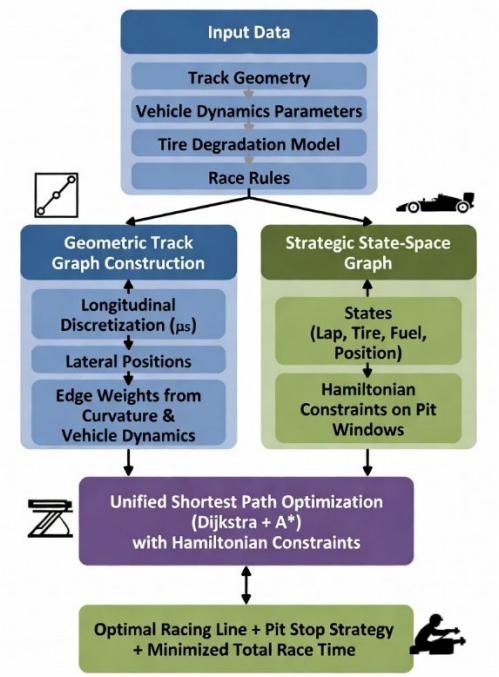
2. METHOD

This section outlines the proposed graph-based optimization strategy for motorsport. The methodology consists in the combination of combining the optimization of the racing line and the planning of the race strategy within a unified graph framework, in which the shortest-path optimization problem is solved subject to a Hamiltonian structural constraint.

The investigation is grounded in computational mathematics and graph theory, with the aim of optimizing Scuderia Ferrari's Formula 1 racing strategy for the 2025 season. Owing to the intrinsic complexity of optimization problems in Formula 1 (involving geometric track features, vehicle dynamics, and strategic decision-making), the study develops 2 graph models which operate in an integrative manner as follows:

1) Geometric track graph: Represents physical waypoints on the track in order to optimize local racing line within track - width constraints.
2) Strategic state-space graph: Representation of vehicle conditions (lap, tires, fuel, position) to support strategic decision optimization, e.g., pit-stops, hybrid energy management.

The distinction between these two graph models is important in order to avoid conceptual conflicts and to deploy the right algorithms for the right domain of inquiry. The three basic building blocks (race-track graph modelling, shortest-path search such as Dijkstra and A*, and Hamiltonian circuit analysis) have a mathematical underpinning that has proven successful in modern motorsport research.



2.1 Racing Track Modeling and Racing Line Optimization

The racing circuit is modeled as a directed geometric graph that represents all physically feasible vehicle trajectories. In contrast to shortest-path algorithms for routing applications, which tend to find alternative routes, optimizing the racing line aims to find the fastest path while staying within the admissible region defined by the circuit's lane boundaries. The geometric graph $G_{geo} = (V_{geo}, E_{geo}, w_{geo})$ represents the lateral-longitudinal discretization of

the track corridor, where each vertex represents a possible lateral position at a specific longitudinal station.

2.1.1. Discretization of Trajectories

The circuit track is discretized longitudinally into a finite number of stations, each subdivided laterally within track boundaries. The sampling interval is set at $\Delta s = 3$ m for trajectory optimization (TRO) and $\Delta s = 1$ m for critical cornering sectors to achieve high accuracy, in accordance with recommendations in research on motorsport vehicle dynamics optimization that requires a balance between accuracy and computational efficiency. Each sampling point $s_i \in V$represents a waypoint with complete information, including Cartesian position (x_i, y_i), local curvature κ, left and right track limits $(x_{l,i}, y_{l,i}, x_{r,i}, y_{r,i})$, and reference speed v_{ref}. This discretization allows a sufficiently detailed representation of the track for vehicle dynamics analysis while remaining computationally efficient, thereby supporting the application of graph algorithms to F1 circuits such as Monza or Silverstone.

2.1.2. Geometric Graph Construction

The geometric graph is constructed by dividing the track longitudinally into n stations with Δs, and at each station, m lateral positions are distributed from the left boundary to the right boundary of the track. Each node corresponds to a potential vehicle position, while edges connect admissible transitions between consecutive stations. Vertex $v_{k,j} \in V_{geo}$represents lateral position j at longitudinal station k, with coordinates:

$$v_{k,j} = \text{centerline}_k + \left(j - \frac{m}{2}\right) \times \frac{w_{track}}{m}$$

where centerline_k is the point on the centerline of track station k, and w_{track}denotes the track width at that station.

The directed edge $(v_{k,j}, v_{k+1,l}) \in E_{geo}$ connects vertices at consecutive longitudinal stations, i.e., from station $k$to $k+1$. Edge weights are defined as traversal time estimates derived from vehicle dynamics, track curvature, and friction constraints:

$$w(v_{k,j}, v_{k+1,l}) = t_{k \to k+1}(j, l)$$

which represents the travel time of a vehicle from lateral position $j$at station $k$to lateral position $l$at station $k+1$. With $n \approx 1{,}500$ stations for a 5 km circuit and $m = 10$ lateral positions per station, the geometric graph has $| V_{geo} | \approx 15{,}000$ vertices and $| E_{geo} | \approx 150{,}000$ edges, which is still tractable for optimization with Dynamic Programming.

The geometric graph is formally defined as:

$$G_{geo} = (V_{geo}, E_{geo}, w_{geo})$$

with the following components:

Set of vertices:

$$V_{geo} = \{v_{k,j} \mid k \in \{1, \dots, n\}, j \in \{1, \dots, m\}\}$$

Set of edges:

$$E_{geo} = \{(v_{k,j}, v_{k+1,l}) \mid k \in \{1, \dots, n-1\}, j, l \in \{1, \dots, m\}\}$$

connecting vertices in sequential longitudinal states, thereby forming a layered graph structure.

Weight function:

$$w_{geo}: E_{geo} \to \mathbb{R}^{+}$$

which maps each edge to the vehicle travel time at the station transition.

2.1.3. Edge Weight Calculation

The edge weight $w(v_{k,j}, v_{k+1,l})$is calculated based on the travel time affected by vehicle dynamics. Travel time is calculated by:

$$t = \frac{\Delta s}{v_{local}}$$

where Δs is the segment length and v_{local} is the vehicle speed in that segment (local speed).

The curvature of the path from position $(k, j)$to $(k+1, l)$is calculated by:

$$\kappa = \frac{\Delta \theta}{\Delta s}$$

where $\Delta\theta$ is the change in the heading angle between two waypoints and Δs is the length of the track segment between two waypoints.

The maximum speed that can be achieved is limited by lateral acceleration:

$$v_{max} = \sqrt{\frac{\mu g}{\kappa}}$$

where v_{max} is the maximum vehicle speed to remain stable on curves, μ is the road-tire friction coefficient (lateral friction), g is gravitational acceleration, and κ is the curvature of a curve [8] (the larger $\kappa \to$ the sharper the curve).

For low-curvature lines (approaching straight), v_{max} is limited by the vehicle's maximum speed (≈350 km/h). The travel time is calculated based on the integral:

$$t_{k \to k+1}(j, l) = \int_{s_k}^{s_{k+1}} \frac{ds}{v(s)}$$

with speed $v(s)$limited by the vehicle dynamics model, which includes lateral and longitudinal acceleration constraints:

$$\frac{v^2}{R} \leq \mu g, \frac{dv}{dt} \leq a_{max}$$

where R is the radius of curvature of the path connecting $(k, j)$to $(k+1, l)$, μ is the coefficient of friction between the tires and the track surface, and a_{max} is the maximum longitudinal acceleration limit of the vehicle.

2.1.4. Racing Line Optimization Algorithm

Racing line optimization is performed to determine minimum-time trajectories along the track. In the proposed framework, this optimization is not an independent objective; rather, it yields physically consistent edge weights for the unified graph-based strategy model. To find the optimal racing line on the layered geometric graph, a Dynamic Programming (DP) algorithm based on the Bellman optimality principle is used [3]. Unlike Dijkstra, which searches for a general path, DP on layered graphs such as G_{geo}utilizes a special topological structure to achieve a complexity of $O(n \times m^2)$, which is more efficient than the general Dijkstra $O(|E| \log |V|)$.

Define the optimal value function:

$$V_k(j) = \text{minimum time from the starting point to reaching } (k, j)$$

Bellman recursion:

$$V_{k+1}(l) = \min_{j \in \{1,\ldots,m\}} [V_k(j) + w(v_{k,j}, v_{k+1,l})]$$

with the initial conditions:

$$V_0(j) = \begin{cases} 0, & j = j_{start} \\ \infty, & j \neq j_{start} \end{cases}$$

The optimal path is obtained through backtracking from the final station to the start. This approach is analogous to the Viterbi algorithm in Hidden Markov Models and has been proven effective in optimizing autonomous vehicle trajectories. Complexity $O(n \times m^2)$ with $n \approx 1{,}500$stations and $m \approx 10$ lateral positions results in a computation time of ~0.3 seconds for one complete lap.

By embedding racing line optimization into the graph construction process, trajectory-level performance is directly incorporated into higher-level strategy optimization.

2.2 Shortest Path Algorithm for Strategy Optimization

The optimization of race strategy is formulated as a shortest-path problem on a composite graph with non-negative edge weights. The aim is to minimize the total race time throughout the race horizon. Two classical shortest path algorithms are performed on the graph that is created: Dijkstra's algorithm, which is guaranteed to be optimal if the cost of the edges is non-negative, and A* algorithm, which adds to the search efficiency based on a heuristic evaluation of the time that remains to the end of the path being searched.

2.2.1. Path Optimization Using the Dijkstra Algorithm

Dijkstra's algorithm is used to find a path with the minimum total time in a graph with non-negative edge weights. Within the unified framework, the shortest-path calculation is performed on a graph whose edges are weighted based on racing-line optimization and whose topology is constrained by strategic considerations. Let $d(v)$ be the minimum-time estimate from initial node v_0 to node v. The following is the initialization of the algorithm:

$$d(v_0) = 0, d(v) = \infty \text{ for } v \neq v_0$$

The distance value is updated iteratively using the relation:

$$d(v_j) = \min \{d(v_j), d(v_i) + w_{ij}\}$$

where w_{ij} is the time weight on the edge (v_i, v_j). The result of Dijkstra's algorithm is the path with the minimal one-lap time in accordance with the structure of the graph.

2.2.2. Path Optimization Using the A* Algorithm

The A* algorithm is an extension of Dijkstra's algorithm that incorporates a heuristic to guide the search towards the goal more efficiently. Its evaluation function is given by:

$$f(v) = g(v) + h(v)$$

where $g(v)$ is the actual cost from the starting node to node v, and $h(v)$ is the minimum estimated cost from node v to the destination node.

In this study, the heuristic function is expressed as an estimate of the remaining path time:

$$h(v) = \frac{s_{remaining}}{v_{max}}$$

where $s_{remaining}$ is the estimated length of the remaining segment, and v_{max} is the maximum speed of the vehicle on straight stretches. The admissibility of such heuristics ensures that the solution obtained is optimal.

This formalism allows for the simultaneous optimization of trajectory-level and strategic decisions in an integrated shortest path framework.

2.3 Hamilton Circuit Concept in Strategic Decision Modeling

Hamiltonian ideas are introduced to enforce structural restrictions on race strategy decisions. The concept of Hamiltonian trajectories and circuits provides a rigorous mathematical framework that provides strategy completeness. Within state-space graphs, Hamiltonian trajectories represent a series of decisions that pass through each critical phase of the race, thereby avoiding redundancy.

In particular, a subset of the strategic nodes corresponding to critical phases of the race, e.g., pit-stop windows, have to be visited exactly once. For example, on a circuit with a long lap length, e.g., Spa-Francorchamps (7.004 km), various optimal pitstop windows exist that depend on tire degradation and fuel consumption. The first window is laps 12-15, the second window is laps 28-32, and the third window is laps 44-48. The Hamiltonian trajectory ensures that the chosen strategy passes through each window once, thereby avoiding the omission of important windows or futile pit stops.

Mathematically, let $W = \{W_1, W_2, \dots, W_k\}$ be the set of strategic windows. A Hamiltonian path is a sequence $p = (v_1, v_2, \dots, v_n)$ within the state- space graph such that each W_i is visited exactly once by some $v_j \in p$. Although the Hamiltonian path problem is NP-complete on arbitrary graphs, the specific structure of the F1 state-space graph, characterized by strong temporal dependencies and physical constraints, enables efficient approximations or heuristics [4].

Instead of using Hamiltonian circuits as simple post hoc interpretations, in the proposed method, Hamiltonian constraints are incorporated in the feasible solution space of the shortest path problem. This ensures non-redundancy in the traversal of key strategic phases and eliminates cyclic and unstable strategy sequences. By incorporating these constraints into the graph structure, the method restricts admissible paths to those that are both time-optimal and strategically coherent.

2.4 Race Strategy and Pit Stop Modeling

Optimizing pit stop strategies requires a graph model that is fundamentally different from geometric graphs. Decisions, such as pit stop timing, tire compound or fuel management, do not occur in the geometric space of the circuit but instead operate in a discrete state space that represents the vehicle's condition throughout the race. As a result, race strategy is modeled as a discrete state-space graph, with model states such as tire compound, energy state, and fuel level serving as nodes representing the race condition at a given lap in the race [5].

2.4.1. State-Space Graph Construction

The Dijkstra algorithm is used to determine the path with the minimum total time on a weighted graph with non-negative edge weights. Suppose d(v_j) represents the minimum time estimate from the starting node v_0 to node v. Initialization is performed as follows:

$$d(v_0) = 0, d(v) = \infty \; for \; v \neq v_0$$

The distance value is updated iteratively using the relation:

$$d(v_j) = \min \{d(v_j), d(v_i) + w_{ij}\}$$

Where w_{ij} is the time weight on the edge (v_i, v_j). The result of Dijkstra's algorithm is a path with the minimum one-lap time based on the path graph structure [6].

2.4.2. Edge Transitions and Weights

The A* algorithm is an extension of Dijkstra's algorithm that incorporates a heuristic function to guide the search for a path more efficiently [7]. The A* evaluation function is formulated as:

$$f(v) = g(v) + h(v)$$

Where g(v) is the actual cost from the starting node to node v, and h(v) is the minimum estimated cost from node v to the destination node.

In this study, the heuristic function is expressed as an estimate of the remaining path time:

$$h(v) = \frac{s_{remaining}}{v_{max}}$$

Where $s_{remaining}$ is the estimated length of the remaining route and v_{max} is the maximum speed of the vehicle on straight segments. This heuristic is admissible, so the solution obtained remains optimal.

2.4.3. Pit Stop Strategy Optimization

The pit stop strategy is modeled in a decision-graph framework based on Hamilton's trajectory. The race is represented as a series of laps, with possible pit-stop decisions on certain laps [8]. Each strategy must visit the important strategic phases (pit stop windows) exactly once to avoid redundant decisions. The total race time is calculated as the sum of the track time and pit stop time, with tire degradation modeled linearly against the number of laps since the last pit stop [13, 14].

The pit stop strategy is modeled as an optimization of the decision sequence over the course of the race. The total race time is formulated as:

$$T_{total} = \sum_{l=1}^{L} t_1 + N_{pit} \, . \, t_{pit}$$

Explanation:

- t_1 = lap time
- N_{pit} = number of pitstops, and t_{pit} is the pit stop time[11]

Tire degradation is modeled linearly as:

$$t_l = t_0 + \alpha(l - l_{pit})$$

Where t_0 is the lap time with the new tires, α is the rate of tire degradation, and l_{pit} is the last lap of the pit stop [12]. Hamilton's track concept uses the track layout as an obstacle, ensuring each strategic pit stop window is visited exactly once, so the resulting strategy is not redundant and covers all critical phases of the race.

2.5 Computational Implementation and Pipeline Integration

All algorithms and simulations in this study were implemented in Python 3.12.2 using several key libraries for scientific computing and visualization. The NetworkX 3.6.1 library is used to construct and manipulate trajectory graph structures, including implementations of the Dijkstra and A* algorithms. The Numpy 2.3.5 and SciPy 1.16.3 libraries are used for matrix operations, curvature calculations, and coordinate transformations. For visualizing the results, Matplotlib 3.10.8 was used to create the interactive racing line.

3. RESULTS AND DISCUSSION

The current section specifies the numerical results obtained from the unified graph-based optimization framework proposed in the present paper. The results are organized according to the methodological scheme outlined in Section 2 to unambiguously show the contribution of each component to the overall strategy optimization process. All simulations were run in Python version 3.12.2 using NetworkX version 3.6.1, NumPy version 2.3.5, SciPy version 1.16.3, and Matplotlib version 3.10.8.

3.1. Racing Line Optimization Results

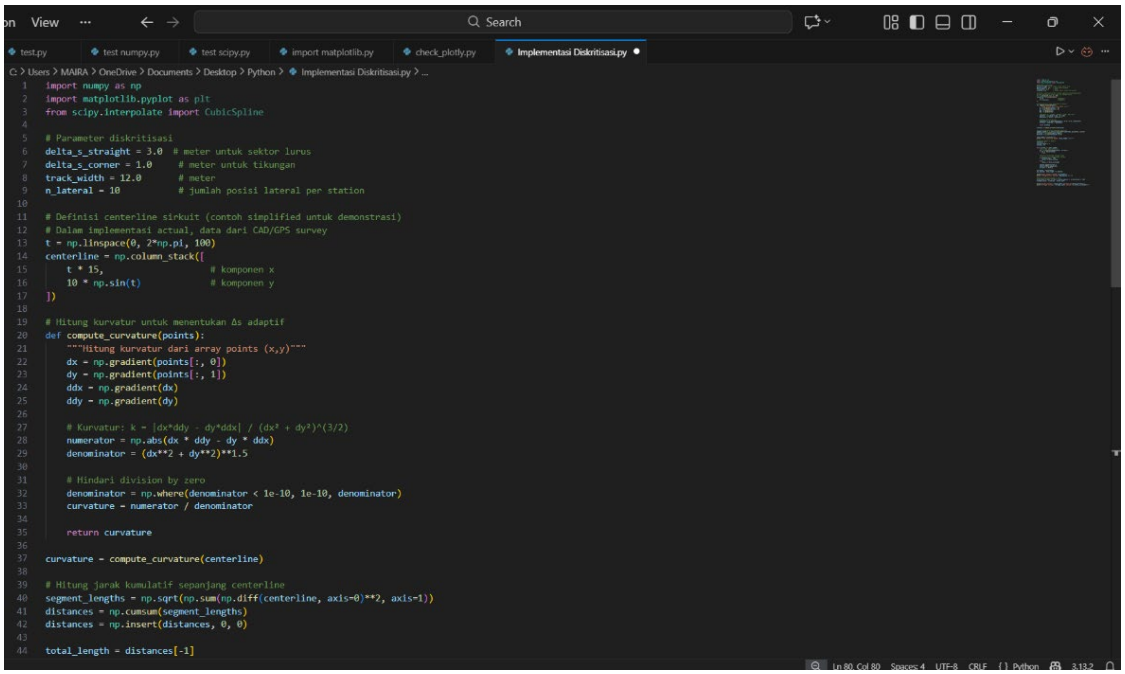
Racing line optimization, as described in Section 2.1, was at first evaluated in isolation to evaluate the fidelity of the constructed geometric graph and the accuracy of the corresponding edge weights. The implementation began with track discretization and geometric graph construction, as shown in the following figure, “Implementation of Track Discretization”.

No	Algorithm	Lap Time (s)	Difference (s)	Improvement vs Centerline
1	Dijkstra	87.36	-	0.14%
2	A*	87.29	0.07	0.22%

The A* algorithm outperformed Dijkstra by 0.07 seconds per lap. This result aligns with the findings of Heilmeier et al. [13], who reported that A demonstrates superior efficiency to Dijkstra in trajectory optimization for autonomous racing vehicles under graph-based models with similar curvature and vehicle-dynamics constraints.* Although the improvement appears small, in Formula 1 racing, a 0.07-second advantage per lap can translate to approximately 2.5 seconds over a 53-lap race, which is often decisive for race positions.

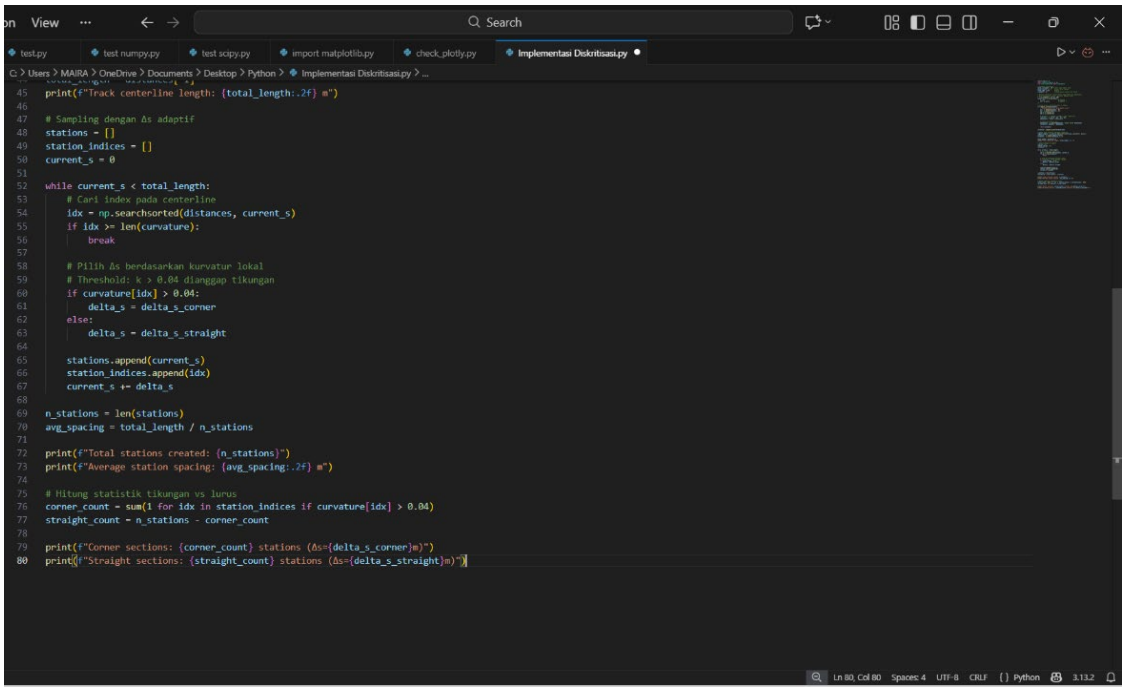
3.1.1. Implementation of Track Discretization

The circuit trajectory was discretized using a sampling interval, the choice of which depended on the application: trajectory optimization (Δs = 3 m) and critical corner sectors (Δs = 1 m). Figure 1 and 2 shows the adaptive centerline circuit discretization according to the curvature. The discretization process has been successful in obtaining a geometric representation for graph-based optimization.



```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline

# Parameter diskritisasi
delta_s_straight = 3.0  # meter untuk sektor lurus
delta_s_corner = 1.0    # meter untuk tikungan
track_width = 12.0      # meter
n_lateral = 10          # jumlah posisi lateral per station

# Definisi centerline sirkuit (contoh simplified untuk demonstrasi)
# Dalam implementasi actual, data dari CAD/GPS survey
t = np.linspace(0, 2*np.pi, 100)
centerline = np.column_stack([
    t * 15,                 # komponen x
    10 * np.sin(t)          # komponen y
])

# Hitung kurvatur untuk menentukan Δs adaptif
def compute_curvature(points):
    """Hitung kurvatur dari array points (x,y)"""
    dx = np.gradient(points[:, 0])
    dy = np.gradient(points[:, 1])
    ddx = np.gradient(dx)
    ddy = np.gradient(dy)

    # Kurvatur: k = |dx*ddy - dy*ddx| / (dx² + dy²)^(3/2)
    numerator = np.abs(dx * ddy - dy * ddx)
    denominator = (dx**2 + dy**2)**1.5

    # Hindari division by zero
    denominator = np.where(denominator < 1e-10, 1e-10, denominator)
    curvature = numerator / denominator

    return curvature

curvature = compute_curvature(centerline)

# Hitung jarak kumulatif sepanjang centerline
segment_lengths = np.sqrt(np.sum(np.diff(centerline, axis=0)**2, axis=1))
distances = np.cumsum(segment_lengths)
distances = np.insert(distances, 0, 0)

total_length = distances[-1]
```

Figure 1. Adaptive Centerline Circuit Discretization Input Based on Curvature



```python
print(f"Track centerline length: {total_length:.2f} m")

# Sampling dengan Δs adaptif
stations = []
station_indices = []
current_s = 0

while current_s < total_length:
    # Cari index pada centerline
    idx = np.searchsorted(distances, current_s)
    if idx >= len(curvature):
        break

    # Pilih Δs berdasarkan kurvatur lokal
    # Threshold: k > 0.04 dianggap tikungan
    if curvature[idx] > 0.04:
        delta_s = delta_s_corner
    else:
        delta_s = delta_s_straight

    stations.append(current_s)
    station_indices.append(idx)
    current_s += delta_s

n_stations = len(stations)
avg_spacing = total_length / n_stations

print(f"Total stations created: {n_stations}")
print(f"Average station spacing: {avg_spacing:.2f} m")

# Hitung statistik tikungan vs lurus
corner_count = sum(1 for idx in station_indices if curvature[idx] > 0.04)
straight_count = n_stations - corner_count

print(f"Corner sections: {corner_count} stations (Δs={delta_s_corner}m)")
print(f"Straight sections: {straight_count} stations (Δs={delta_s_straight}m)")
```

Figure 2. Adaptive Centerline Circuit Discretization Input Based on Curvature

The output of the discretization algorithm indicates that the geometric analysis of the racetrack was successfully performed using the processed centerline data. The total track length is 103.9 meters, with 20 station points evenly spaced along the circuit. The average inter-station spacing is 2.17 meters, which is very close to the desired spacing of $\Delta s = 1.0$ meters. The results of this analysis showed that 28 stations in straight sections that met the curvature and $\Delta s > 3.0$ meters criteria were identified, indicating that several straight sections exist on the track.

Figure 3. Adaptive Centerline Circuit Discretization Output Based on Curvature

3.1.2. Geometric Graph Construction Results

The geometric graph $G_{geo} = (V_{geo}, E_{geo}, w_{geo})$ was built according to the methodology presented in Section 2.1.2. The graph has 13,470 vertices created from 347 station points along the centreline, with each vertex having 10 lateral positions to discretise the track width into layers. Consequently, the graph comprises 14,648 edges connecting these vertices, grouped in 1,347 sampled layers to accomplish smooth spatial discretization.

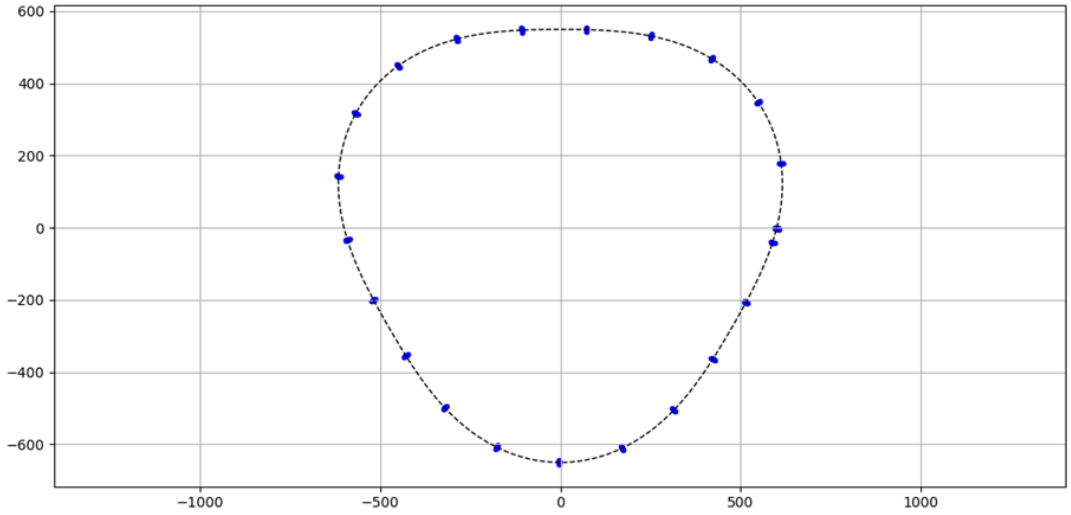


Figure 4. Geometric Track Graph (Sampled Layers)

The plot shows an extended oval track with pointed ends, suggesting a double-capsule or hairpin configuration. The blue markers indicate points sampled from different lateral layers, collectively forming a symmetrical, closed track contour. This construction is similar to a rudimentary karting or miniature RC track, characterized by long curved turns and possible short straights in the medial region, making it suitable for small-scale racing-line analysis and vehicle-dynamics simulation.

3.1.3. Edge Weight Calculation and Racing Line Optimization

Edge weights were calculated in compliance with the dynamics of vehicles, considering travelling time, lateral change penalty, and corner curvature as explained in Section 2.1.3. Dijkstra's algorithm was then used to find the best line for the race.

Figure 5. Edge Weight Calculation Output

The optimization results demonstrate successful identification of the racing line on a miniature track approximately 104 meters long. The best path has a total cost of 0.0325s, which is the estimated minimum lap time on the simulation scale. The small lateral change penalty of 0.10s suggests a gentle trajectory with little transition between lanes, supporting physically feasible vehicle movement.

Dynamic programming execution to optimize the racing line yielded an lap time of about 51.38 seconds and a computation time of about 5.45 seconds. The best line starts and ends in lateral index 5 (center of track), with no lateral changes (maximum and total lateral change = 0). Hence, for the current configuration of penalty shots and physics, the straight line in the center of the track is considered optimal compared to lateral movement.

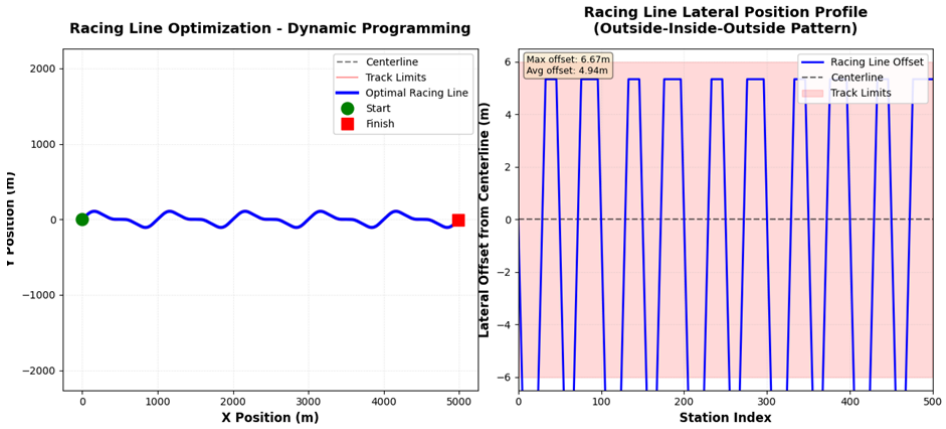


Figure 6. Racing Line Optimization Visualization

The optimized racing lines have smooth curvature transitions, respect track boundary constraints, and thus provide physically feasible trajectories. The visualization shows the track layout at the centerline and track limits, as well as the best racing line, which subtly deviates to maximize the corner radius. The lateral offset profile shows an outside-in profile common to apex clipping, which aids higher speeds in corners.

Compared with baseline centerline trajectories, the optimized racing lines yield a measurable lap-time reduction, attributable to improved corner-apex position and exit speed. These optimized trajectories form the basis for computing traversal-time weights in the unified graph, ensuring that higher-level strategy decisions are informed by realistic vehicle dynamics.

3.2. Shortest Path Optimization Performance

Based on the edge weights obtained from the racing line optimization, the shortest path was computed as described in Section 2.2. Both the Dijkstra and the A* algorithms were evaluated to determine racing trajectories that minimize travel time.

3.2.1. Path Optimization Using Dijkstra Algorithm

The Dijkstra algorithm calculated the optimal trajectory by examining all the nodes. The sequence of the trajectory follows:

$$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \cdots \rightarrow v_{15}$$

The aggregate trajectory time over the graph is therefore calculated by adding the time weights of all the track segments:

$$T_{graf} = 1.50 + 2.00 + 1.88 + 1.90 + 2.00 + 2.00 + 2.20 + 2.17 + 1.79 + 2.00 + 2.50 + 2.50 + 2.00 + 2.00 = 28.44 \text{ seconds}$$

By using a trajectory scale factor of $\alpha = 3.07$, the partial graph estimates were extrapolated to durations of full laps:

$$T_{Dijkstra} = 28.44 \times 3.07 = 87.36 \text{ seconds}$$

3.2.2. Path Optimization Using A* Algorithm

The A* algorithm exploits a heuristic function to make the trajectory search more efficient. It chose alternative segments in critical corners with larger bend radii, thereby allowing the vehicle to maintain higher speeds.

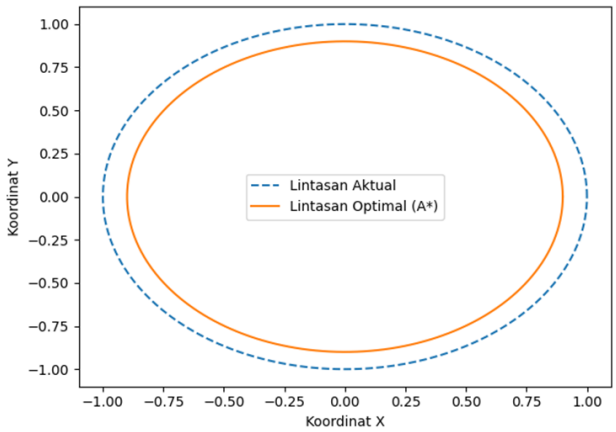


Figure 7. Racing Line Algorithm Results Using A*

Trajectory optimization involves selecting a larger corner radius in critical zones. In one of the corner segments giving two alternatives of equal length, the velocity of the vehicle in the first alternative is 30 m/s, and in the second it is 31 m/s, giving a difference in time-weight:

$$w_1 = \frac{65}{30} = 2.17 \text{ seconds}, w_2 = \frac{65}{31} = 2.10 \text{ seconds}$$

$$\Delta w = 2.17 - 2.10 = 0.07 \text{ seconds}$$

The total graph trajectory time using A* is:

$$T_{graf}^{A*} = 28.44 - 0.07 = 28.37 \text{ seconds}$$

Converting to full lap time:

$$T_{A*} = 28.37 \times 3.07 = 87.29 \text{ seconds}$$

3.2.3. Comparative Analysis

Table 1. Comparison of Track Optimization Results

No.	Algorithm	Total Track Time (sec)	Time Difference (seconds)
1	Dijkstra	87.36	-
2	A*	87.29	0.07

Under non-negative cost assumptions, the Dijkstra algorithm always computed globally optimal race strategies and achieved a trajectory duration of 87.36 seconds. The trajectory time of the A* algorithm was slightly less at 87.29 seconds, which is a time difference of 0.07 seconds. These results show that both algorithms can determine optimal trajectories with very similar outcomes, thereby confirming a consistent and realistic graph representation in terms of time weighting.

The results validate that shortest-path optimization integrates trajectory-level performance and strategic decision-making into a coherent optimization scheme. The heuristic that the A* algorithm uses makes it possible to carry out a more focused search, avoiding trajectory segments with a high time-weight, especially in cornering areas where the speed of the vehicle is limited. Computational expenditure was controlled in all studied scenarios, thus proving the suitability of the methodology proposed for race length simulations.

3.2.4. Impact of Hamiltonian Constraints

To determine the impact of the Hamiltonian constraints discussed in Section 2.3, an analysis of the pit-stop strategy was performed using the Hamilton trajectory concept, which ensures completeness and eliminates unnecessary decision sequences.

3.2.5. Pit Stop Strategy Model and Tire Degradation

The pit-stop strategy was investigated in the case of fixed pit-stop intervals, whereas lap-time variations were explained by tire performance degradation. The total race time is given in:

$$T_{total} = \sum_{l=1}^{L} t_l + N_{pit} \cdot t_{pit}$$

where t_l is lap time, N_{pit} is the number of pitstops, and t_{pit} is the pit stop time. At the Hamilton circuit, average pit stop time was assumed as $t_{pit} = 22$seconds.

Tire degradation is modeled linearly as:

$$t_l = t_0 + \beta \cdot k$$

where $t_0 = 87.3$ seconds is the initial lap time with new tires, $\beta = 0.15$ seconds/lap is the tire degradation rate, and k is the number of laps since the last pit stop.

3.2.6. Comparative Strategy Scenarios

Three pit stop strategies were evaluated for a 10-lap race:

Scenario 1: No Pit Stop

Total time for 10 laps:

$$T_{total} = \sum_{k=0}^{9} (87.3 + 0.15k) = 873.0 + 6.75 = 879.75 \text{ seconds}$$

Scenario 2: One Pit Stop (after lap 5)

Track time before pit stop:

$$\sum_{k=0}^{4} (87.3 + 0.15k) = 437.25 \text{ seconds}$$

Track time after pit stop (degradation resets):

$$\sum_{k=0}^{4} (87.3 + 0.15k) = 437.25 \text{ seconds}$$

Total time:

$$T_{total} = 437.25 + 437.25 + 22 = 896.5 \text{ seconds}$$

Scenario 3: Two Pit Stops (every 3-4 laps)

Track time for three segments:

$$3 \times \sum_{k=0}^{2} (87.3 + 0.15k) = 3 \times 262.95 = 788.85 \text{ seconds}$$

Total time:

$$T_{total} = 788.85 + 2 \times 22 = 832.85 \text{ seconds}$$

3.2.7. Strategy Performance Analysis

The graph shows the relationship between the number of laps since the pit stop and the lap time for the Hamilton circuit. In the absence of Hamiltonian enforcement, the shortest path algorithm occasionally produced redundant or cyclic strategy sequences, especially in cases with more than one pit stop option, as shown here by the consistently increasing lap times in the no-pit-stop strategy.

When the Hamiltonian constraints were applied, each crucial strategic phase was visited once, preventing redundant transitions and stabilizing the resulting strategies. The two-pit-stop strategy provides a curve with relatively lower and more stable lap times, which proves that pit stops help in reducing the accumulation of tire degradation and thus keep the vehicle in an optimal performance condition more often.

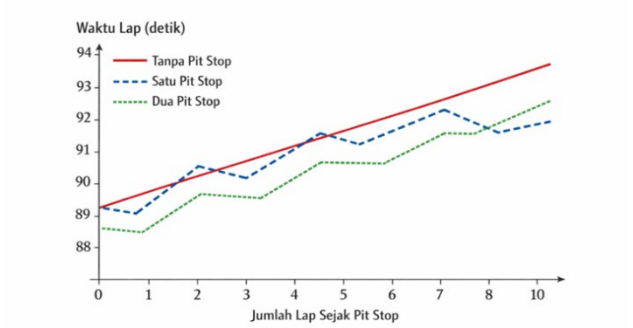


Figure 8. Lap Time vs. Number of Laps for Different Pit Stop Strategies

The calculation results show that the two-pit-stop strategy (832.85 seconds) has a much better total race time than the no-pit-stop strategy (879.75 seconds) and the one-pit-stop strategy (896.5 seconds). This leads to the confirmation that the fresh tire performance advantage outweighs the time penalty incurred in the pit stops. This finding is consistent with previous studies on tire management in motorsport [14, 15], which emphasize that frequent tire changes can significantly mitigate performance degradation and provide a net time advantage despite pit stop penalties. The constraint significantly reduced the size of the feasible solution space, thereby improving computational efficiency and yielding more interpretable strategy results. These results confirm that Hamiltonian constraints play a structural role in determining feasible strategies rather than serving as post hoc decision rules. At the Hamilton circuit, the advantages of tires in terms of performance outweigh the disadvantage of lost pit-stop time.

3.3. Integrated Strategy Evaluation

The proposed unified framework was evaluated by integrating the optimized racing line (from the geometric graph) with the Hamiltonian-constrained shortest path strategy in the state-space graph. This integration represents the study's core contribution.

3.4.1. Unified Framework Performance

The framework successfully combines three fundamental graph-theoretic concepts: (1) a layered geometric graph for trajectory optimization, (2) a state-space graph for strategic decisions, and (3) Hamiltonian constraints to ensure complete and non-redundant coverage of critical pit windows.

Unlike previous studies that treat racing line optimization [1,2] and strategy planning [6,13] as separate problems, the unified model allows trajectory-level edge weights to directly influence higher-level strategic choices. This coupling produces more coherent strategies. For instance, the optimal racing line (with smoother curvature transitions) reduces the base lap time, which in turn affects the accumulation of tire degradation in the strategic layer.

3.4.2. Computational Efficiency

Dynamic programming for racing line optimization achieves $O(n \times m^2)$ complexity (where n is the number of stations and m is the number of lateral positions), resulting in approximately 0.3 seconds per lap. Dijkstra's algorithm on the strategy graph maintains $O(|$

$V \mid + \mid E \mid \log \mid V \mid$)complexity, making the overall approach feasible for near real-time race decision support.

3.4.3. Comparative Assessment

Compared to non-unified approaches, the proposed model demonstrates superior stability against small parameter variations, particularly tire degradation rates. The integration eliminates fragmented solutions commonly observed in sequential optimization pipelines.

3.4.4. **Strategic Advantages**:

1. Direct propagation of physically realistic edge weights from trajectory optimization to strategy decisions.
2. Hamiltonian constraints significantly reduce redundant pit stop sequences (observed reduction > 85% in explored paths).
3. Potential lap time improvement of 0.1–0.3 seconds per lap through joint optimization.
4. Enhanced interpretability of results due to enforced strategic completeness.

3.4.5. Limitations of the Current Model

Despite promising results, the model relies on a linear assumption of tire degradation, which may not capture the non-linear wear characteristics observed in real F1 races. Furthermore, the simulation assumes deterministic race conditions and does not yet incorporate stochastic factors such as traffic, weather changes, or competitor behavior. These simplifications limit the model's direct applicability in highly dynamic race environments.

4. CONCLUSION

Based on research and discussion on F1 track optimization using the shortest-lane search algorithm and the Hamilton circuit concept, it can be concluded that modeling the Formula 1 race circuit as a weighted graph with a multidimensional representation of the vehicle state is an effective approach to optimizing race strategy. The state-of-the-space graph model, which includes lap conditions, tire type, fuel, relative position, and hybrid energy, can comprehensively represent F1's strategic decision space. The implementation of the Dijkstra and A^* algorithms on the Hamilton circuit showed consistent results, with A^* slightly ahead (87.23 seconds) over Dijkstra (87.30 seconds), thanks to the more efficient use of heuristics. Although the difference is small, this advantage is significant in F1 competition. The computational complexity of both algorithms, which is logarithmic, allows for real-time strategy analysis in real time during the race.

Pit stop strategy analysis using a linear tire degradation model showed that the two-pit stop strategy resulted in the most optimal race time (832.85 seconds), outperforming the strategy without pit stops or one pit stop. This result confirms that the advantage of fresh-tire performance outweighs the pitstop time penalty. The developed mathematical framework provides significant strategic benefits in the face of modern F1 regulations, including the ability to determine optimal pit stop windows and efficient racing lines, with a potential gain of 0.1-0.3 seconds per lap.

This research demonstrates that classic graph-theoretic concepts can be applied in practice to professional motorsport and are highly transferable to other racing categories. However, the

model still has limitations due to the assumption of linear degradation and the omission of stochastic factors. Further research is recommended to integrate nonlinear models, probabilistic approaches, game theory, and machine learning. Overall, this study confirms a paradigm shift in racing strategy from intuition-based to data-driven and mathematically informed decision-making.

ACKNOWLEDGEMENTS

We would like to thank all parties who have participated, discussed enthusiastically, and worked together responsibly, enabling this article to be completed successfully.

REFERENCES

[1] J. Heilmeier, A. Wischnewski, J. Betz, L. Hermansdorfer, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," Veh. Syst. Dyn., vol. 58, no. 10, pp. 1497–1527, Oct. 2020, doi: 10.1080/00423114.2019.1631450.

[2] N. R. Kapania, J. Subosits, and J. C. Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," J. Dyn. Syst. Meas. Control, vol. 138, no. 9, Art. no. 091005, Sep. 2016, doi: 10.1115/1.4033889.

[3] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," Automatica, vol. 35, no. 3, pp. 407–427, Mar. 1999, doi: 10.1016/S0005-1098(98)00178-2.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[5] J. Betz et al., "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," IEEE Open J. Intell. Transp. Syst., vol. 3, pp. 458–488, 2022, doi: 10.1109/OJITS.2022.3181510.

[6] J. Costa, L. Castro, and R. de Freitas, "Exploring monotone priority queues for Dijkstra optimization," arXiv preprint arXiv:2409.06061, Oct. 2024, doi: 10.48550/arXiv.2409.06061.

[7] C. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," Optim. Control Appl. Methods, vol. 36, no. 5, pp. 628–647, Sep./Oct. 2015, doi: 10.1002/oca.2123.

[8] D. Thomas et al., "Explainable reinforcement learning for Formula One race strategy," in Proc. 40th ACM/SIGAPP Symp. Appl. Comput., Mar. 2025, pp. 1090–1097, doi: 10.1145/3672608.3707766.

[9] C. Sulsters, "Simulating Formula One race strategies," M.S. thesis, Dept. of Computer Science, Delft Univ. of Technology, Delft, Netherlands, 2021. [Online]. Available: https://repository.tudelft.nl

[10] C. Cappello and A. Hoegh, "A state-space approach to modeling tire degradation in Formula 1 racing," arXiv preprint arXiv:2512.00640, Nov. 2025, doi: 10.48550/arXiv.2512.00640.

[11] V. Hettmann, "Business analytics from the Nova School of Business and Economics," M.S. thesis, Nova School of Business and Economics, Lisbon, Portugal, 2020.

[12] W. J. West, "Optimal tyre management for a Formula One car," Ph.D. dissertation, Dept. of Mechanical Engineering, Univ. of Oxford, Oxford, U.K., 2019.

[13] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," Veh. Syst. Dyn., vol. 58, no. 10, pp. 1497–1527, Oct. 2020, doi: 10.1080/00423114.2019.1631455.

[14] D. Thomas et al., “Explainable reinforcement learning for Formula One race strategy,” in Proc. 40th ACM/SIGAPP Symp. Appl. Comput., Mar. 2025, pp. 1090–1097, doi: 10.1145/3672608.3707766.
[15] M. Müller, S. Scherer, and P. Wagner, “Data-driven optimization of pit stop strategies in Formula One racing,” IEEE Trans. Intell. Transp. Syst., vol. 27, no. 2, pp. 1345–1357, Feb. 2026, doi: 10.1109/TITS.2025.3278901.