Comparative Analysis of Dynamic Programming and A* Algorithms in Optimizing Tire Strategy and Pit-Stop in F1 Racing

Stanislaus Ardy Bramantyo - 18223057
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: stanislausardy@gmail.com , 18223057@std.stei.itb.ac.id

***Abstract*— Deciding when to pit and which tire to use is one of the biggest strategy choices in a Formula 1 race. We model the tire and pit stop problem as a shortest path problem on a directed acyclic graph with non-negative edge weights, and solve it with two algorithms, Dynamic Programming with backward induction and A* with an admissible heuristic, so any difference comes from the algorithms and not the problem. Every parameter of the lap time model and the pit time loss is fit from real race data through the FastF1 library, using the 2023 Bahrain Grand Prix driven by Max Verstappen over 57 laps. Fit on 49 clean green flag laps, the model reaches an R^2 of 0.84 and a mean absolute error of 0.19 s, with a pit loss of 24.42 s. It gives a one stop optimum, start on Soft and pit on lap 29 to Hard, for a modeled race time of 5586.39 s. Both solvers return this same optimum, while A* reaches it with about 18 % fewer states and less peak memory. A scaling test shows the state space growing quadratically for both, with A* keeping a bounded constant lead. The one stop optimum is 13.14 s faster than Verstappen's actual two stop race, which shows what the simple single car model leaves out rather than a driver mistake.**

Keywords—A* algorithm, dynamic programming, Formula 1, pit stop strategy, tire degradation mode, shortest path.

I. Introduction

Formula 1, commonly abbreviated as F1, is the most prestigious international racing for single-seater formula racing cars [1]. 22 cars from 11 different teams race around a circuit, reaching speeds of up to 350 km/h, while competing to cross the checkered flag first. F1 isn't just about racing, but a combination of brilliant engineering and strategic decision-making.

Although F1 race strategy is shaped by numerous variables, including weather conditions, track temperature, and humidity, this research focuses specifically on tire management and pit stop strategy due to their significant impact on race outcomes. Here, teams and drivers must find the right balance between staying on track as their tires age and performance drops, or entering the pit lane, accepting the time loss of a pit stop in exchange for a fresh set of tires, or possibly a different compound [2]. While the FIA does not impose a minimum number of pits stops for a standard dry race, teams and drivers typically complete two stops over the course of the race. As a result, tire management and pit stop strategy become critical factors in a team's race planning.

Such problems are commonly optimized through computational algorithms, including A* search and dynamic programming. In this context, dynamic programming breaks the problem down into smaller, overlapping subproblems, solving and storing the result of each to build toward an optimal overall solution. A* formulates the problem as a shortest-path search over a graph, using an evaluation function $f(n) = g(n) + h(n)$ to efficiently identify the optimal path. Here, $g(n)$ represents the actual cost from the start, while $h(n)$ represents the heuristic estimate to the goal.

This paper does three things. First, both DP and A* are applied to the same tire and pit-stop problem formulation to ensure a fair comparison between the two algorithms. Second, all parameters of the optimization formulation are estimated from real race data using the FastF1 library and regression on clean laps. Third, the two algorithms are tested and compared on both solution quality and runtime, with the findings discussed to better understand how each algorithm performs in this problem. To ensure a more consistent analysis, this research focuses only on the 2023 Bahrain Grand Prix, specifically on driver Max Verstappen across all 57 laps.

II. Theoretical Foundation

A. Dynamic Programming

Dynamic Programming (DP) is an algorithm for solving optimization problems by breaking a problem into overlapping subproblems, solving each subproblem exactly once, and storing its solution for reuse [3]. DP can be applied efficiently when a problem exhibits two properties [3]:

1. Optimal Substructure, where an optimal solution to a problem is built from optimal solutions to its subproblems.
2. Overlapping Subproblems, where the same subproblem recurs multiple times, making its result worth storing rather than recalculating.

The formal foundation for this is Bellman's Principle of Optimality, which states that in an optimal sequence of decisions, all remaining decisions must be optimal relative to the state produced by the first decision [3]. This establishes the foundation of the Bellman recurrence. In a minimization problem defined over a set of states, each associated with a set of possible successor actions, the optimal value function *V(s)* represents the minimum cost from state *s* to a terminal state, and satisfies.

$$V(s) = \min [c(s, s') + V(s')], \qquad V(s_{goal})=0, \qquad (1)$$

With a note that *c(s, s')* is the transition cost to a successor *s'*. Equation (1) is essentially solved by backward induction where the terminal states have a value of 0, and other states are calculated backward from states closer to the goal.

B. Shortest Path on a Directed Acyclic Graph

A Directed Acyclic Graph, commonly abbreviated as DAG, is a directed graph with no cycles, meaning its vertices admit a topological ordering in which every edge always points toward a later vertex [3]. Its nodes can be lined up in a single forward order in which every arrow points from an earlier node to a later node.

This matters here because the tire-strategy state space is essentially a DAG in itself. Every decision advances the lap index from *k* to *k+1* and never returns to an earlier lap, or in other words there can't be any cycles to ever be formed. Because the graph only moves in a forward direction, a single well-defined best path through it exists, and DP algorithm can help compute it. That exact answer also serves as the *ground truth* against another search algorithm, which is A*.

C. A* Algorithm

A* is an informed best-first search for a minimum-cost path, introduced by Hart, Nilsson, and Raphael [4]. A* is essentially a priority queue that always expands the node with the smallest *f(n)* value.

$$f(n) = g(n) + h(n) \qquad (2)$$

With notes being *g(n)* is the actual cost from the root or the start lap to *n* and *h(n)* is the heuristic value where essentially means the estimated cost from *n* to the goal node [4]. In some cases, heuristic values can be counted from the straight-line value from node *n* to the goal node, though here it is quite different.

Using a heuristic value in an equation gives the consequences to validate if each heuristic value is admissible or not, by using this equation

$$h(n) \leq h^*(n) \qquad (3)$$

With notes *h*(n)* is the true optimal cost from node *n* to the goal node. This validation helps to guarantee A* can find an optimal path later on, if it's supported by the admissibility of each the heuristic value [4].

D. Lap-Time F1 Race Model

In a typical dry F1 race, lap time is primarily determined by tire condition and fuel. The common race-simulation approach models it as a sum of a few components [5].

$$t_{lap} = t_{base}(k) + \delta(k) \cdot a + \phi \cdot \ell, \qquad (4)$$

With notes being $t_{base}(k)$ is the fresh-tire pace of compound *k*, $\delta(k)$ its per-lap degradation rate of the tire, *a* the tire age for the current set, ϕ is the fuel-effect coefficient, and ℓ being the lap index. The term $\delta(k) \cdot a$ models degradation, commonly taken as linear in age (as the tire gets older, it gets slower too), while $\phi \cdot \ell$ accounts for how lap times will improve as fuel mass decreases and the track surface changes (more tire residue) throughout the race.

The parameters are estimated from race lap data, with safety car laps and in/out laps excluded from it, consistent with the approach taken in most race-simulation research and the state-space framework for tire degradation in F1 [6]. Within FIA regulations, teams are required to use at least two different tire compounds within a dry race. Because the decisions of when to pit and which compounds to use are made lap by lap, the problem naturally becomes a discrete sequential optimization, and this is exactly the kind of problem that DP and A* are built to handle.

E. FastF1 as the Source of Data

FastF1 is an open-source Python library that gives programmatic access to official Formula 1 data such as lap times, telemetry, tire compound information, track status, and pit entry and exit times [7]. Here, it provides the real-world data used to fit the parameters in Equation (4).

III. Problem Modeling

This section sets up the tire and pit-stop problem as a single discrete sequential optimization, which both the DP and A* algorithms use without modification. Solving the same problem with both algorithms is what makes the later comparison as fair as possible. The formulation follows the classical operations research pattern, which includes *state*, *decision*, *transition*, *cost*, *constraint*, and *objective* [3].

A. State, Decision, and Transition

A race of *N* laps is formulated as a single forward sequence of decisions, on for each lap. Each situation in that sequence is captured by a *state s*, which can be defined as

$$s = (\ell, k, a, U), \qquad (5)$$

where $\ell \in \{1, \ldots, N+1\}$ is the lap about to be run, *k* is the dry compound currently fitted, *a* is the tire age of the current set, and *U* is the set of all compounds used so far in the race.

The first three components of Equation (5) determine the lap-time of Equation (4). When a race begins, we start with a determined state which is (1, k, 0, $\{k\}$) with k being the choice of compounds that the team and driver use to start with.

On every lap the driver and team chooses one of two decisions, which together will form the successor function of the state graph:

1. Stay Out, where drivers continue to run another lap on the current set. The compound is unchanged and the tire ages by one lap, giving the successor $(\ell+1, k, a+1, U)$
2. Pits, where drivers enter the pit lane and fit in a fresh set of compound d. The lap is completed on these fresh tires, d is added to U, and the successor is $(\ell+1, k, 1, U \cup \{d\})$

Because every decision advances the lap index by 1 (from ℓ to $\ell+1$) and never returns, the resulting state graph is considered to be a *Directed Acyclic Graph* (DAG) described before. This property is what allows both DP and A* algorithm to compute the same optimal path over it.

B. Per-Lap Cost and Objective Function

Each edge of the graph carries a cost equal to the time of the lap it represents. Using what have been defined before in Equation (4),

$$t_{lap}(k,a,\ell) = t_{base}(k) + \delta(k) \cdot a + \phi \cdot \ell \quad (6)$$

A Stay Out edge costs exactly the lap-time, whilst a Pits edge runs the lap on a fresh set and adds the pit time loss p_{loss}.

$$c(s,s') = t_{lap}(k, a, \ell), \quad \text{for Stay Out} \quad (7)$$

$$c(s,s') = t_{lap}(d, 0, \ell) + p_{loss}, \quad \text{for Pits} \quad (8)$$

The objective is to minimize the total race time all around, which can be equated as the sum of each per-lap costs along the chosen sequences of decisions for each lap.

$$T = \sum c(s_\ell, s_{\ell+1}) \quad (9)$$

Since every edge cost is a lap-time and is therefore non-negative, the problem is a shortest-path problem on a scope of DAG with non-negative weights. With this, we can state the setting in which backward-induction DP and A* with an admissible heuristic are guaranteed to return the same minimum.

C. Feasibility Constraint

Under FIA regulations, each car in a dry Formula 1 race must use at least two different dry compounds [2], [5]. This rule is reflected directly in the U component of the state, where a state qualifies as a valid goal only when the race is finished and at least two unique compounds have been used.

A sequence that reaches the end of the race on a single compound would never satisfies FIA regulations hence it isn't considered to be a valid solution. Placing the rule in the goal test, rather than checking it after the fact, lets both solvers prune single-compound paths as part of the search and ensures both algorithms operate on the same logic.

D. Data Source and Lap Cleaning

All parameters of Equation (4) are estimated from real race data so that the optimization rests on genuine pace rather than an imaginary number. The data are pulled with FastF1 library. For this particular paper, we focuses only on the 2023 Bahrain Grand Prix, driver Max Verstappen, N = 57 laps. With consistent data hope that every result can be reproduced later. Defining a fixed event and driver gives a helping hand on the set of compounds that are used.

A sequence that reaches the end of the race on a single compound would never satisfies FIA regulations hence it isn't considered to be a valid solution. Placing the rule in the goal test, rather than checking it after the fact, lets both solvers prune single-compound paths as part of the search and ensures both algorithms operate on the same logic

Raw race data are usually noisy, so only clean green-flag racing laps are kept for this particular paper. A lap is retained only if it satisfies all of the following constraint:

1. Track status is green, which indicates there is no safety or virtual safety car.
2. It is neither a pit in-lap nor pit out-lap.
3. It has a valid recorded lap-time in the data.

E. Parameter Estimation and Pit-Loss Calibration

The compound-specific intercepts $t_{base}(k)$, the degradation slopes $\delta(k)$, and the shared fuel coefficient ϕ are obtained by a single ordinary least-squares regression over all clean laps. Each clean ap contributes to one row of the design matrix A and one observed lap-time in b. For a given lap, the feature row records a 1 in the intercept column of the fitted compound to capture its base lap time, the current tire age in that compound's age-slope column to capture its degradation rate, and the lap index in a single shared column to capture the effect of decreasing fuel mass.

Each compound gets its own base lap time and degradation rate, while the fuel and track-evolution term is shared across all compounds because it reflects how the car and track change over the race, not how the tire behaves [5], [6].

The pit time loss is measured from the same session rather than assumed. It is the average extra time that pit-affected laps take compared to the median clean lap time. For the case-study race this comes out to approximately 24.4 s, with a fallback of 22.0 s used only when no usable pit laps are available.

The model is then checked against the clean laps it was trained on, giving $R^2 \approx 0.84$ and MAE ≈ 0.19 s per lap. This is a reasonable fit for a simple linear model on real race data and small enough to support building the optimization on top of it.

IV. Implementation and Application

This section describes how the shared problem model is implemented in code and then applied to the race study case. The code is written in Python and keeps the problem definition in one place, so that both the Dynamic Programming (DP) and A* solvers consume the same state graph, cost function, and feasibility test. This is done to uphold fairness between these two algorithms.

A. System Architecture and Data Pipeline

The program is divided into modules with distinct roles, keeping the data, the model, the search, and the measurement cleanly separated:

- data_loader, which pulls the race session with FastF1 library, cleans the laps, counts the race length N, and computes the pit time loss using Equation (9).
- model, which fits the lap-time model of Equation (4) by least squares and provides the per-lap cost used by both algorithm.
- state_graph, which holds the single shared definitions of states, the successor function, and the goal test.
- solver_dp and solver_astar, which are the two algorithm, each operating only through state_graph and model.
- benchmark, which runs both algorithms under identical conditions and records the cost, number of states explored, time to solve, and peak memory.
- plots, which renders the optimal strategy, the cumulative race time, and the scaling curves.

The data pipeline runs once per execution. FastF1 library loads the race session and caches it locally so that repeated runs always use the same data. Because the model object is shared, the two solvers see numerically identical edge cost.

B. Shared State-Graph Implementation

The state is implemented exactly as stated in Equation(5), represented as the ordered group (ℓ, k, a, U), with the set of used compounds U kept as a sorted fixed sequence so that each state can be looked up efficiently by both search algorithms. For each compound available at the start of the race, one initial state of the form $(1, k, 0, \{k\})$ is created.

The successor function covers the two decisions at each lap, Stay Out and Pit, and attaches the edge cost of Equation (6) to each one. Given a state (ℓ, k, a, U), it produces the following next states:

- 1 Stay Out edge to $(\ell + 1, k, a + 1, U)$ at the cost $t_{lap}(k, a, \ell)$
- 1 Pit edge per available compound d to $(\ell + 1, k, a + 1, U \cup \{d\})$ at the cost $t_{lap}(k,a,\ell) + p_{loss}$.

The goal test follows Equation (8) directly: a state counts as a goal only when $\ell = N+1$ and $|U| \geq 2$. Since both the successor function and the goal test are defined in this single module, DP and A* are guaranteed to explore the same graph and follow the same rules, with neither algorithm needing to re-implement them.

C. Dynamic Programming Solver

The DP algorithm implements the Bellman recurrence of Equation (1) using top-down backward induction, where each state is computed once and its result stored for reuse. The function V(s) represents the minimum remaining time from state s to a feasible finish and is computed recursively. A goal state returns 0; a state that reaches $\ell = N+1$ without satisfying the two-compound rule is marked infeasible and returns $+\infty$; otherwise the algorithm evaluates every successor and takes

$$V(s) = \min [c(s,s') + V(s')] \tag{10}$$

Both the computed value and the best decision at each state are saved in a lookup table, so each state is only ever evaluated once. The first time a state is visited its value is computed and saved; any later visit simply reads the stored result in constant time, which is the overlapping-subproblems property from Section II-A in practice. Once all start states are evaluated, the best start state is identified and its stored decisions are followed forward to reconstruct the optimal strategy. A counter tracks how many states are newly evaluated, giving the work measure reported in Section IV-E. Following Equation (2), the running time is $O(|S| \cdot d)$, where $|S|$ is the number of distinct states and d is the branching factor (here $d = 1 + |\text{compounds}|$).

D. A* Solver

The A* algorithm carries out the best-first search of Equation (3) on the same state graph used by DP. It keeps a binary-heap priority queue keyed by f, a g-score map holding the best known cost from a start state to each state, and a closed set so that a state finalized once is not explored again later. At each step, the algorithm selects the most promising unexplored state, defined as the one with the lowest f score, checks whether it is a goal state and returns immediately if so, and otherwise evaluates every reachable successor, updating the cost of any successor for which a cheaper path has been found and re-inserting it into the search. The work measure is the number of states that the algorithm selects and fully evaluates, serving as the direct counterpart to the state count recorded for DP.

The heuristic estimates how much time is left in the race from any given state and is guaranteed never to overestimate it. Taking $t_{\min}$ as the fastest lap time achievable across all compounds, tire ages, and lap indices, and ℓ as the next lap in state s, the estimated remaining cost can be formulated as

$$h(s) = (N - \ell + 1) \cdot t_{\min}, \qquad t_{\min} = \min t_{\text{lap}}(k, a, \ell) \tag{11}$$

Because every remaining lap must cost at the very much least $t_{\min}$ and a pit stop only adds a non-negative penalty, $h(s)$ can never exceed the true remaining cost $h^*(s)$. Knowing that, then every $h(s)$ must be admissible. Because of the heuristic value being admissible, A* algorithm is guaranteed to return the same optimum as DP.

E. *Application to the 2023 Bahrain Grand Prix*

The 2023 Bahrain Grand Prix that have been stated before are applied with the current pipeline. Of the 57 laps, 49 of them we're categorized as clean laps after the filtering process. For the pit time loss, it was estimated from Equation (9) to give a value of $p_{\text{loss}} \approx 24.42$ s.

1) Fitted lap-time model. The single least-squares fit produced the parameters in Table 1. The fit explains most of the lap-to-lap variataion, with a coefficient of determination being $R^2 \approx 0.838$ and the mean absolute error being MAE $\approx$ 0.193 s each lap.

TABLE I. FITTED LAP-TIME MODEL PARAMETERS

Parameter	SOFT	HARD
Base pace t_{base} (s)	98.001	97.920
Degradation δ (s/lap)	0.0452	0.0479
Fuel coefficient ϕ (s/lap)	-0.0353	-0.353

As a note, there are a few shared fit qualities that's been defined beforehand, that would include R^2, MAE, and p_{loss}.

2) Optimal Strategy. Solving the model gives an optimal strategy of a start on soft compound and make a single pit stop on lap 29 to a hard compound (one-stop race), for a total modeled race time of **5586.39 s**. The optimal strategy and the corresponding race time accumulated lap by lap are shown in Fig. 1 and Fig. 2. For reference, Verstappen's actual race was a two-stop strategy, being soft compound to start, change to a new fresh of soft compound for the first pit stop, and ended with a hard compound.

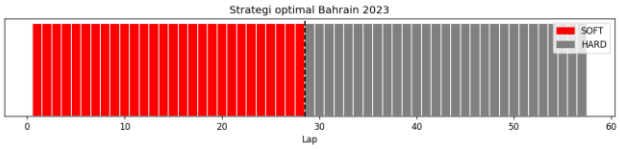

Fig. 1. Optimal one-stop strategy for the 2023 Bahrain GP (Verstappen, 57 laps). Each lap appears as a colored bar (red = Soft, gray = Hard), and the dashed line marks the pit stop on lap 29. The car starts on Soft and switches to Hard for the rest of the race.

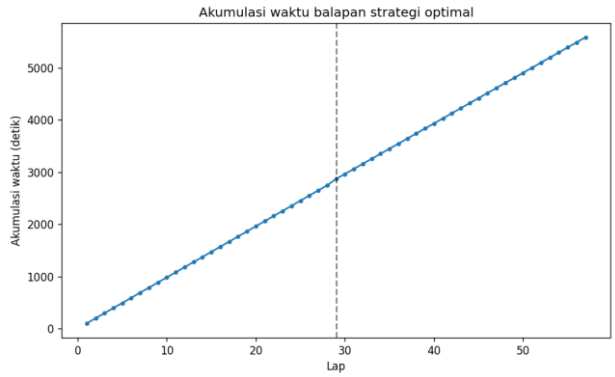

Fig. 2. Modeled race time of the optimal strategy, shown lap by lap. The dashed vertical line marks the lap-29 pit stop, which appears as a small jump in the curve due to the pit time loss ($p_{\text{loss}} = 24.42$ s). The total race time at the finish is 5586.39 s.

3) Solver Comparison. Both algorithms were tested on the same 57-lap model. As the theory in Section II-C predicts, they produced the same optimal cost and differed only in how much of the state graph each one examined. Table II reports the cost, number of states examined, wall-clock time, and peak memory.

TABLE II. SOLVER COMPARISON ON THE CASE-STUDY RACE

Solver	Cost (s)	States	Time (s)	Memory (KB)
DP	5586.390	6386	0.3302	1872.5
A*	5586.390	5235	0.2370	1476.7

The two costs match to within numerical tolerance, confirming in practice that DP and A* find the same optimal solution. A* arrives at that optimum having examined around 18% fewer states (5235 compared to 6386), because its admissible heuristic (Equation (11)) guides the search toward the goal rather than expanding the full graph the way DP does.

4) Scaling experiment. To see how this gap changes with problem size, both algorithms were run on the same model truncated to several race lengths. Table III and Fig. 3 show that the number of states grows with race length for both solvers, while A* consistently examines fewer states than DP at every size tested.

TABLE III. FITTED LAP-TIME MODEL PARAMETERS

Laps	SOFT	HARD
10	182	163
20	762	652
30	1742	1454
40	3122	2584
50	4902	4031
57	6386	5235

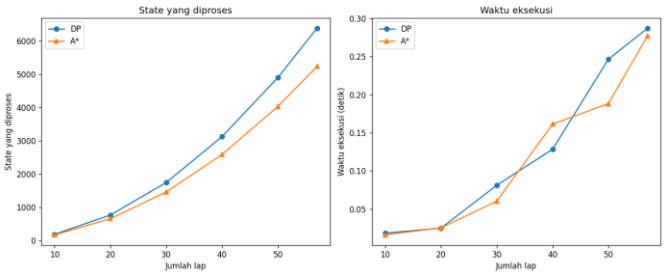

Fig. 3. How the two solvers scale with race length. Left: number of states examined; right: execution time. A* (triangles) examines fewer states than DP (circles) at every race length tested, while both return the identical optimal cost.

Together, these results show that the implementation reproduces the theoretical guarantee of identical optima from both solvers, while also quantifying the practical difference between them. Why A* expands fewer states, the gap between the modeled optimum and Verstappen's real two-stop race, and the limitations of the simple lap-time model are all interpreted in Section V.

V. Analysis and Comparison

This section interprets the results of Section IV rather than simply restating them. It works a few questions, which are did the two algorithms agree, as the theory of Section II predicts; why did A* examine fewer states; how does that advantage behave as the race grows longer; and how does the model's optimum compare to the strategy Verstappen drove. It closes with an account of what the simple model leaves out, setting up the conclusions of Section VI.

A. Empirical Confirmation of the DP–A* Equivalence

The central theoretical claim of this paper, that on the same *Directed Acyclic Graph* (DAG) with non-negative edge weights, Dynamic Programming (DP) by backward induction and A* with an admissible heuristic return the identical optimum (Section II-C), is borne out exactly. Both solvers report a total modeled race time of **5586.390 s**, matching to within numerical tolerance ($<10^{-6}$ s, Table II). They also reconstruct the same decision sequence: start on Soft and pit once on lap 29 to Hard. This is not a coincidence of the implementation but a consequence of two design choices from Sections III–IV: both solvers share a single state graph and cost function, and the A* heuristic of Equation (11) is provably admissible. The agreement therefore validates the implementation (the solvers contain no divergent logic) and empirically reproduces the optimality guarantee, meaning A* can be trusted to return the true optimum and not merely a good path [3], [4].

Because the two answers are identical, the remainder of the comparison concerns not solution quality, since there is no quality gap to discuss, but purely the computational work each algorithm performs to reach that answer.

B. Search Efficiency: Why A* Examines Fewer States

On the 57-lap race, A* examined 5235 states against DP's 6386, a reduction of approximately 18% (Table II). The cause lies in the role of the heuristic of Equation (3), $f(n) = g(n) + h(n)$. DP, as backward induction, assigns a value to every reachable state in the DAG; it has no notion of a promising state and therefore evaluates the entire reachable graph. A*, by contrast, orders its frontier by f and expands the most promising states first, so any state whose optimistic total cost already exceeds the best-known feasible finish is never popped. The admissible lower bound $h(s) = (N - \ell + 1)\, t_{min}$, with $t_{min} \approx$ **95.91 s**, supplies that ordering: it allows A* to recognize early that the remaining laps cannot cost less than t_{min} each, and to prune those states accordingly.

That said, the saving should not be overstated. The heuristic is deliberately conservative: it sets aside tire degradation, the fuel trend, and the mandatory second-compound rule, and floors each lap at only the single cheapest time physically possible. An admissible heuristic that is this loose still guarantees optimality, but its pruning power is weak, putting it much closer to Dijkstra's algorithm than to a perfectly informed search [4], [5]. The 18% reduction is therefore moderate by design: the heuristic steers A* in the right direction, but does not steer it hard.

Two further points put the headline number in perspective. First, wall-clock time is the least reliable of the three measures. State count and peak memory are deterministic and give the same result every run, whereas the times in Table II (DP 0.33 s, A* 0.24 s) are sensitive to interpreter scheduling. They are also affected by A*'s per-state overhead, since every relaxation requires a heap push/pop and a heuristic evaluation, and this partly cancels out the gain from examining fewer states. Repeated runs show the two times falling within measurement noise of each other, so the number of states examined is the fairer measure of actual algorithmic work. Second, A*'s lower peak memory (1476.7 KB versus 1872.5 KB, about 21% less) makes sense given that it finalizes fewer states. Part of this saving is offset by the open-heap entries A* holds in memory at any given time, but the overall footprint is still smaller than DP's full value table.

C. Scaling Behavior with Race Length

Truncating the same model to shorter races (Table III, Fig. 3) shows how the two solvers scale. For both, the number of states grows quadratically with race length: each state is a (lap, tire-age) pair combined with a compound choice, and since tire age can range up to the lap index, the reachable state space is

$$|S| = O(N^2), \tag{12}$$

This means DP's cost is $O(|S| \cdot d) = O(N^2 d)$ from Equation (2). The measured DP state counts (182, 762, 1742, 3122, 4902, 6386 at N = 10, 20, 30, 40, 50, 57) show near-constant second differences, the empirical signature of a quadratic, which confirms Equation (12).

The more telling quantity is the ratio of A* to DP states, which grows as the race lengthens:

TABLE IV. Solver Comparison on the Case-Study Race

Laps	DP states	A* states	A*/DP ratio	Saving
10	182	163	0.896	10.4%
20	762	652	0.856	14.4%
30	1742	1454	0.835	16.5%
40	3122	2584	0.828	17.2%
50	4902	4031	0.822	17.8%
57	6386	5235	0.820	18.0%

The saving grows from about 10% on a short race to about 18% at full distance and shows clear signs of leveling off near 18% rather than continuing to rise. This points to the heuristic

giving A* a constant-factor advantage over DP in the long run, not a difference in how the two algorithms grow: both remain $O(N^2)$ in states examined, and A* simply explores a fixed fraction (about 0.82) of what DP does. This is the practical consequence of the point made in Section II-C that a tighter heuristic prunes more: with a loose bound, the benefit is real but modest. On very short races, the cost of initializing all start states leaves little room for pruning, which explains why the saving is smallest at low N.

D. Optimal Strategy versus Verstappen's Actual Race

The model identifies a **one-stop** race as optimal (Soft to lap 28, followed by Hard to the flag), whereas Verstappen in fact ran a **two-stop** strategy (Soft, Soft, Hard, with stints of approximately 14, 22, and 21 laps). Evaluating his actual strategy under the same fitted cost function yields a modeled time of **5599.534 s**, which is **13.14 s** slower than the one-stop optimum of 5586.390 s. The source of this gap is straightforward. An additional pit stop incurs a fixed cost of $p_{loss} \approx 24.42$ s, while the only compensating benefit is the degradation avoided by running fresher tires, at rates of $\delta \approx 0.045$ to 0.048 s per lap of tire age (Table I). Across a stint, this accumulated degradation does not offset the 24.42 s incurred by the extra stop, and the model therefore favors a single stop.

Two features of the fitted model explain which one-stop plan comes out ahead. The two compounds have almost identical fresh-tire pace (Hard 97.920 s, Soft 98.001 s) and similar degradation, so the model barely distinguishes between them on compound order: starting on Hard and finishing on Soft scores 5586.394 s, just 0.004 s behind the chosen Soft-then-Hard plan. Because the compounds are nearly symmetric and the cost rises linearly with both tire age and fuel, the optimizer ends up splitting the race almost evenly, placing the single stop close to the midpoint (lap 29 of 57). The result is therefore robust in value but only weakly pinned down in its details, a sensitivity worth starting plainly.

The 13.14 s gap should not be read as a sign that Verstappen made a mistake. It is the distance between an idealized deterministic model and a real race, and it reflects what the model leaves out rather than any driver error. Two-stop strategy buys things the model simply cannot see: protection against thermal degradation and graining over one long stint, the freedom to respond to or launch an undercut against rivals, lower exposure if a Safety Car or a tire problem comes up, and more pace in hand when running in traffic. The model strips the race down to deterministic pace, fuel, and pit loss, and the leftover 13.14 s is exactly the price of those simplifications, which the next subsection sets out in detail.

E. Limitations of the Model

The results above are only as reliable as the lap-time model in Equation (4), and that model is deliberately simple. Its main limitations are:

- **Deterministic and single car.** The model has no Safety Car, Virtual Safety Car, rain, or red flag. Yet these are exactly the events that tend to drive real strategy, since they cheapen a pit stop or bunch the field. It is also single-car, with no rivals, which puts the undercut and overcut battles and the track position behind a real two-stop race entirely outside its scope [2], [6]. This is the biggest contributor to the Section V-D gap.
- **Linear degradation.** Equation (4) assumes degradation grows linearly with tire age. Real tires rarely behave that way, since they often hold steady for a while before their grip drops off rapidly, and thermal and graining effects add further curvature that a straight line cannot capture. Because of this, the model can understate how costly very long stints really are, which biases it toward fewer stops [5], [6].
- **Cleaning and in-sample evaluation.** Because the fit relies only on clean green-flag laps, it ignores exactly the Safety-Car and traffic laps that have the biggest effect on real strategy. The reported R^2=0.838 and MAE=0.193 s is also in-sample, meaning they were measured on the same laps used for fitting. As a result, they tell us how well the model fits the data, not how well it predicts laps it has not seen.
- **Aggregate pit loss**. The pit loss $p_{loss} \approx 24.42$ s in Equation (9) is just one number, averaged across the whole session, that rolls pit-lane time, the stop, and in/out-lap effects together. This means it stays the same regardless of traffic or where the car sits in the pit lane.
- **Algorithmic, rather than modeling, scope of the heuristic**. Finally, the A* heuristic in Equation (11) is admissible but loose, which limits the search saving to a modest constant factor. A tighter admissible bound, for instance one that accounts for unavoidable tire degradation or the mandatory use of a second compound, would prune more of the search space without changing the optimum, and is a natural direction for future work.

VI. Analysis and Comparison

This paper compared Dynamic Programming (DP) and A* search algorithm on a single, shared formulation of the Formula 1 tire and pit-stop problem, with every parameter of the lap-time model fitted from real race data using FastF1. By putting both algorithms on the same state graph, the same cost function, and the same feasibility test, the study narrowed the comparison down to the one thing that legitimately differs between them, namely how much of the search space each must examine, and tested it on the 2023 Bahrain Grand Prix (GP) case study. Three conclusions follow, corresponding to the paper's three stated aims.

A. Summary of Findings

1) The DP and A equivalence holds empircally.* Running on the same Directed Acyclic Graph (DAG) with non-negative edge weights, backward-induction DP and A* with an admissible heuristic returned the identical optimum. Both reported a modeled race time of 5586.390 s, agreeing to within $< 10^{-6}$ s, and reconstructed the same decision sequence, which was to start on Soft and make a single pit stop on lap 29 to

Hard. This confirms, on real data, the theoretical guarantee that the two algorithms are simply two routes to one answer, and it validates the implementation, since the solvers share no divergent logic [3], [4].

2) What A gains from its heuristic is less work, not a bettern answer*. Because the two solutions are identical, the only thing left to compare is effort. A* reached the optimum after examining 5235 states against DP's 6386 (about 18 % fewer), using roughly 21 % less peak memory (with 1476.7 KB vs. 1872.5 KB). Because Equation (11) provides an admissible per-lap lower bound, A* can prune any state whose optimistic cost already exceeds the best feasible finish, whereas DP evaluates the whole reachable graph and cannot prune at all. In the scaling experiment, the state count grew quadratically, $|S| = O(N^2)$, for both solvers, while the A*/DP state ratio flattened toward roughly 0.82 as the race lengthened. The heuristic therefore buys a bounded constant-factor advantage, not a change in growth order, and that modest saving is exactly what a deliberately loose (though admissible) heuristic costs. Wall-clock time turned out to be the least reliable measure, since A*'s per-state heap and heuristic overhead pushed the two times within run-to-run noise. The fair measure of algorithmic work is the deterministic state count, not the clock.

3) The data-driven optimum is a one-stop race. Fitted from 49 clean green-flag laps ($R^2 = 0.838$, mean absolute error 0.193 s; pit loss $p_{loss} = 24.42$ s), the model favored a one-stop strategy, while Verstappen actually ran a two-stop race on Soft, then Soft, then Hard. Under the same fitted cost function, his real strategy comes out to 5599.534 s, 13.14 s slower than the one-stop optimum. The extra stop adds a fixed 24.42 s, and that cost is not recovered by the small degradation it saves, which amounts to only about 0.045 to 0.048 s per lap of tire age. This gap is best read not as a driver error but as a measurement of what the model leaves out. A two-stop race buys protection against non-linear thermal degradation on a long stint, the ability to undercut or overcut rivals, and lower risk under a Safety Car, none of which a deterministic, single-car model can capture. Because the two compounds ran at nearly symmetric pace, the optimum was robust in value but only weakly determined in its details, with HARD-then-SOFT coming in just 0.004 s worse. This is a sensitivity worth keeping in view.

Taken together, the findings meet the paper's goal. On a fairly shared model, DP and A* a are interchangeable in correctness and differ only in search effort, and on this problem A* offers a real but bounded efficiency gain. The central algorithmic claim is about the relationship between DP and A* on a shared model, and it holds regardless of how realistic the underlying lap-time model is.

B. Limitations and Future Work

The conclusions about the algorithms are solid. The conclusions about the specific Bahrain strategy are bounded by the simplicity of the lap-time model, and those bounds point directly to future work:

- **Tighten the heuristic.** The A* bound of Equation (11) uses only the single cheapest possible lap as the per-lap floor, ignoring unavoidable degradation and the mandatory second compound. A tighter admissible bound that accounts for these would prune more aggressively and improve the constant factor discussed earlier, all without changing the optimum, which makes it the most natural algorithmic extension [4], [5].

- **Model stochastic events**. Safety Cars, Virtual Safety Cars, and weather changes are the events that most often drive real strategy, since they cheapen a pit stop or bunch up the field. Adding them would move the model from deterministic optimization toward the stochastic and online planning that the literature already explores [2], [6].

- **Non-linear degradation**. The linear degradation term in Equation (4) could be replaced with a nonlinear model, such as a cliff or thermal formulation. This would remove the bias toward fewer stops and let the fit capture very long stints more faithfully [5], [6].

- **Multi-car interaction.** Extending the single-car model to account for rivals, track position, and the undercut and overcut would let the optimizer value a two-stop strategy for the competitive reasons it actually serves, which could close much of the 13.14 s gap [2], [6].

APPENDIX

https://github.com/SArdyBramantyo/Astar_DP_F1_PitStop_TireSet_Comparison

ACKNOWLEDGMENT

I would acknowledge all my friends that took this course with me this semester, even though there were ups and lots of downs, it's been a blessing to the very least been in this situation. I would also acknowledge my parents and my coworkers that have given a few opportunities to take some time and work on this paper within work hours and family time. And lastly, I would like to acknowledge God that have been helping, indirectly of course, and have let me finish this work and finish these past few weeks that's been filled with exams, work, round trips from Jakarta to Bandung, Indonesia's instable political and economic state, and lots of down. Yet here I am writing the last part of my paper.

REFERENCES

[1] Formula 1. “Everything You Need to Know about F1.” *Formula 1*, 2025, www.formula1.com/en/latest/article/drivers-teams-cars-circuits-and-more-everything-you-need-to-knowabout.7iQfL3Rivf1comzdqV5jwc.

[2] O. F. C. Heine and C. Thraves, “On the optimization of pit stop strategies via dynamic programming,” *Central European Journal of Operations Research*, vol. 31, no. 1, pp. 239–268, Jun. 2022, doi: 10.1007/s10100-022-00806-4.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009. K. Elissa, “Title of paper if known,” unpublished.

[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson, 2021.

[5] A. Heilmeier, M. Graf, and M. Lienkamp, “A Race Simulation for Strategy Decisions in Circuit Motorsports,” in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018.

[6] C. Cappello and A. Hoegh, “A state-space approach to modeling tire degradation in Formula 1 Racing,” arXiv.org, https://doi.org/10.48550/arXiv.2512.00640

[7] FastF1, "FastF1 documentation." [Online]. Available: https://docs.fastf1.dev/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Jakarta, 19 Juni 2026

Stanislaus Ardy Bramantyo dan 18223057