

Article

Performance Analysis of Energy-Efficient Path Planning for Sustainable Transportation

Dimitris Georgiadis ^{1,*} , Konstantina Karathanasopoulou ^{1,*} , Cleopatra Bardaki ¹ , Ilias Panagiotopoulos ² , Ioannis Vondikakis ¹ , Thalys Paktitis ¹  and George Dimitrakopoulos ¹ 

¹ Department of Informatics and Telematics, Harokopio University of Athens, 9 Omirou Str., 17778 Athens, Greece; cleobar@hua.gr (C.B.); ivondi@hua.gr (I.V.); thalispak@hua.gr (T.P.); gdimitra@hua.gr (G.D.)

² Informatics and Telematics Department, Harokopio University of Athens, 17778 Tavros, Greece; ipanagio@hua.gr

* Correspondence: dgeorgiadis@hua.gr (D.G.); kkarathanasopoulou@hua.gr (K.K.)

Abstract: Optimizing path planning for energy efficiency is critical for achieving sustainable vehicular transportation. This paper presents a novel framework for evaluating the impact of path planning algorithms (PPAs) on energy consumption within a simulated environment. We leverage the CARLA simulator to conduct a comparative analysis between the widely used A* and a Hybrid Genetic Algorithm (HGA) across diverse vehicular scenarios. This investigation aims to quantify the influence of PPA selection on vehicle energy expenditure, enabling data-driven optimization for energy minimization. We leverage an offline energy estimation model to further streamline the comparison of the two PPAs. Extensive simulations are employed to demonstrate the efficiency and adaptability of the proposed framework. The findings contribute to the development of energy-efficient path-planning strategies, promoting advancements toward sustainable transportation systems.

Keywords: HGA; A*; Carla; path-finding; sustainability; energy efficiency



Citation: Georgiadis, D.; Karathanasopoulou, K.; Bardaki, C.; Panagiotopoulos, I.; Vondikakis, I.; Paktitis, T.; Dimitrakopoulos, G. Performance Analysis of Energy-Efficient Path Planning for Sustainable Transportation. *Sustainability* **2024**, *16*, 4963. <https://doi.org/10.3390/su16124963>

Academic Editors: Sotirios Kontogiannis and Theodosios Gkamas

Received: 30 April 2024

Revised: 24 May 2024

Accepted: 31 May 2024

Published: 11 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The transport sector is a significant contributor to global energy consumption and greenhouse gas emissions. As urban populations continue to grow, the need for sustainable and efficient transportation solutions becomes increasingly critical. In this context, energy-efficient vehicle path planning for sustainable transportation emerges as a promising approach for reducing reliance on fossil fuels and promoting the adoption of electric vehicles. By optimizing routes for individual vehicles or fleets, such frameworks can significantly decrease energy consumption, leading to a more sustainable transportation system.

The benefits of energy-efficient path planning [1] extend beyond environmental considerations. By suggesting shorter or less congested routes, these frameworks can enhance user experience by reducing travel time and potentially lowering battery charging costs. This is particularly relevant for electric vehicles with limited range, where efficient route planning can significantly improve usability and reduce “range anxiety” for drivers. The rapidly evolving field of path planning algorithms (PPAs) [2] offers tremendous potential to revolutionize urban mobility, paving the way for a transport future characterized by sustainability, efficiency, and user-friendliness.

This paper addresses a crucial gap in the existing research on energy-efficient path planning for sustainable vehicle transport. While current research often focuses on theoretical efficiency metrics like shortest path length, this study investigates the practical implications of path planning choices on energy consumption, proposing a novel framework for analyzing the impact of different PPAs on energy usage in a simulated real-world environment. In particular, two well-established PPAs will be taken into consideration: A* [3] and Hybrid Genetic Algorithm (HGA) [4]. Both algorithms strive to deliver optimal or near-optimal

solutions, expanding the applicability of path-planning techniques. In this context, every path-planning method operates within a defined state space. As such, the paper applies these PPAs to a detailed map provided by a simulated environment (Carla) [5], containing all necessary coordinates and, therefore, concomitantly exploring their effectiveness. Let it also be noted that the proposed solution allows for the integration of an energy estimation methodology alongside vehicle-specific characteristics, providing a more comprehensive understanding of the energy efficiency of different path-planning strategies.

Since the conceptualization of our work revolved around the simulator, we opted to include all the available maps in a scenario that orders a specified vehicle to execute and test the optimal path as provided by the PPAs and also evaluated by the energy model. Thus, the vehicle executes an unknown route from an input depot and destination, always based on the road topology of each map, validating how the different road network changes the energy demands, as well as increasing the complexity with each different map, challenging the PPAs. The vehicle is obliged to follow fundamental rules comprising the road network. For instance, the vehicle has to stop when in front of a traffic light, or a stop sign as well as respect other vehicles present in the environment. Lastly, the physical controls are adjusted to accustom the different altitudes between roads and speed limits based on the type of city, which are integrated into the energy estimation model.

We note that comparing HGA with a heuristic algorithm such as A* is a nuanced task as they are fundamentally different and suited for distinct application tasks. Hence, it is less common to find direct comparisons between HGAs and A* in the literature. In cases where a vehicle aspires to find the shortest route, the A* is preferred for its high efficiency; nonetheless, in a scenario involving minimizing distance, consumption, and time HGAs provide a more comprehensive solution. Thus, while comparing HGA and A* is challenging, it can be appropriate under certain conditions. In this work, we experimented with both PPAs, keeping in mind their characteristics, and transformed the problem into a common challenge, optimizing the distance. In the end, though, we considered that it would be more productive to only compare their performance given different scenarios (in this case, maps) and evaluate how they performed when combined with an energy estimation model, which requires a priori knowledge of the route and the distance between nodes. Finally, since we believe our solution to be a test bed for Carla and PPAs, we assumed that having both a metaheuristic and heuristic to be the most comprehensive approach.

The structure of the paper is as follows. In the next section, Section 2, we provide a comprehensive analysis of the state of the art for path planning and energy estimation algorithms, and we present the software solution. Section 3 contains a detailed overview of our workflow for our approach, providing essential information regarding our implementations of the PPAs, the offline energy estimation model, and vehicle modulation. Section 4 showcases the results of our experimentations, focusing on the comparison between the two PPAs leveraging the energy estimation, and finally, in Section 5, we conclude our work and present our next steps.

2. Related Work

Energy efficiency has become a critical aspect of sustainable transportation networks, prompting significant research efforts in developing optimized path planning algorithms (PPAs) that minimize vehicle energy consumption. Existing studies have explored various approaches to achieve this goal. Energy-efficient path planning for vehicles has garnered significant research attention. Studies exist that explore real-time traffic integration for dynamic route optimization [6], and research has been conducted on incorporating vehicle specifics, like engine type and battery capacity, into pathfinding models to improve efficiency [7].

There are several literature references regarding the A* algorithm. Most of them try to identify the optimal route for Highly Automated Vehicles (HAVs) and Unmanned Aerial Vehicles (UAVs) [8]. While A* is a classic algorithm, research continues to explore its applications and improvements. As an example, there are surveys of search algorithms for

pathfinding in Video Games [9]. On the other hand, Hybrid Genetic Algorithms (HGAs) elevate the capabilities of traditional Genetic Algorithms (GAs) by incorporating additional optimization techniques. GAs, inspired by biological evolution, operate on a population of potential solutions (paths) that undergo selection, crossover, and mutation. Through these iterative steps, the population evolves towards increasingly optimal solutions. HGAs go beyond this by integrating complementary methods like local search algorithms. These local search algorithms act as a fine-tuning mechanism during selection, allowing for the more precise refinement of promising paths. This combined approach proves particularly advantageous in path planning problems where the optimal route might be hidden within a complex landscape, potentially escaping the grasp of a basic GA [10,11].

In general, HGA associates the global search attribute with other methods, such as local search or heuristic rules, thus transforming into a balanced and efficient solution enhanced to explore the entirety of the solution space and distinguish the best feasible solutions. When compared with a fundamental metaheuristic, such as Simulated Annealing (SA) [10], the HGA allows a broader search space exploration, in contrast to a single solution reliant on the SA, making it prone to finding and getting stuck with local optima. Additionally, HGAs are suited for multi-objective optimization, where they can simultaneously optimize a set of objectives in an efficient manner [12]. In [13], the authors presented a novel approach of HGA to optimize the Vehicle Routing Problem (VRP) with time windows. Their algorithm consistently performed well across diverse problem instances, where the results matched or improved best-known solutions (BKS). The authors note that the low deviations from the BKS ensure the algorithm's practical application in an industrial environment. While Simulated Annealing (SA) [10] and Ant Colony Optimization (ACO) [14] are valuable tools for uncertain or complex problems, when prioritizing finding the absolute optimal solution in a well-defined search space (like pathfinding), then A* and Hybrid Genetic Algorithm (HGA) are likely the best choices (Table 1). Their strengths in deterministic environments and guaranteed convergence (for HGA) make them ideal for scenarios where an optimal outcome is critical.

Table 1. Path planning algorithms (PPAs) comparison. With the color red we denote the Low Level of Appropriateness, with orange is the Medium and with green is the High. We use this Level of Appropriateness to highlight the utility of each algorithm for our application based on the metrics included.

Algorithms vs. Comparison Factors	Simulated Annealing (SA)	Hybrid Genetic Algorithm (HGA)	Ant Colony Optimization (ACO)	A*
Problem Type	Medium	High	Medium	High
Convergence and Runtime	Low	High	Medium	High
Parallelization	High	Medium	High	Medium
Memory Requirements	Medium	Medium	Low	High

The last common metaheuristic approach for path planning is the Particle Swarm Optimization (PSO). The fundamental PSO is capable of solving the path planning challenge but is prone to significantly large routes, poor global search ability, and local development ability. The performance is also significantly hardened in dynamic, complex environments including but not limited to obstacles, thus requiring more resources on environmental adaptation and path accuracy [15]. When juxtaposed to the HGA [16], the inferiority of PSO was outlined with the GA producing a precise optimum design, with only half iterations needed by PSO in a complex optimization challenge.

When focusing on real-world applications, the HGA can provide better solutions than the fundamental A* algorithm, especially where complex optimization challenges are to be solved. For instance, HGAs are more effective in engineering optimization problems, as they are able to efficiently explore composite, multi-dimensional search spaces

in order to determine the global optima, as is the case in [17], where HGA significantly improved solution accuracy and reliability compared to traditional methods. As stated, heat exchangers are critical industrial components practically found in a wide range of industrial applications. However, their complicated design criteria and high manufacturing costs correspond to a necessity for optimizing their design, lowering the economic cost. Additionally, HGA is more suitable for difficult scheduling problems, such as airline crew scheduling [18], and also proves to be a good choice for real-time application systems, such as scheduling a multiprocessor for a machine vision system [19].

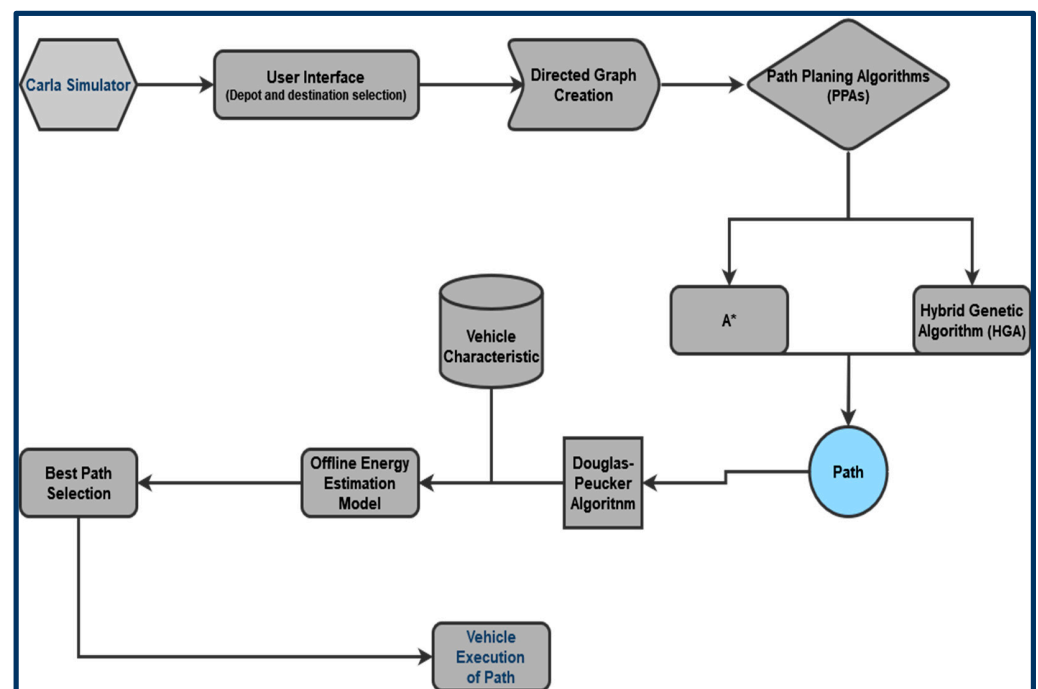
Estimating vehicle energy consumption involves two main approaches: modeling and data-driven. Modeling uses physics principles to create simulations, while data-driven methods rely on real-world data to build statistical models. Researchers often combine these for better accuracy. Current research focuses on making models more comprehensive and enabling real-time energy use estimation, ultimately improving fuel efficiency, EV range prediction, and vehicle design [20]. Beyond just vehicles, research conducted in smart grids is pioneering stochastic and distributed optimal energy management for Active Distribution Networks (ADNs) integrated with buildings. This approach tackles network energy optimization by considering factors like building dynamics and occupant behavior [21]. In this paper, we have leveraged a battery electric vehicles (BEVs) algorithm for estimating a journey's total energy consumption before departure. This algorithm boasts a standard mean deviation of up to 10%, offering a reliable pre-trip energy prediction tool. This contribution complements existing route optimization algorithms by incorporating the energy factor into route selection, paving the way for a more comprehensive approach to finding the most suitable driving route [22].

CARLA (CAR Learning to Act) is a prominent tool for research in autonomous vehicles. Researchers leverage CARLA's diverse sensor suite to simulate realistic perception tasks like object detection and lane recognition [23]. Furthermore, CARLA's flexibility enables the creation of standardized benchmarks for evaluating self-driving algorithms and the design of custom scenarios for targeted testing in specific conditions [24]. Finally, the CARLA simulator has been used in several routing research works [25], much like the one presented in this paper, making it a valuable asset for advancing the development and effectiveness of autonomous vehicles.

This work builds upon these efforts by proposing a novel framework where the user can choose between an A* algorithm and an HGA and simultaneously can see the impact of these path-planning algorithms on energy consumption in a simulated environment. This analysis aims to provide a more comprehensive understanding of how algorithm choice influences energy usage in vehicles. A* is a widely used and well-understood path-planning algorithm known for its efficiency in finding the shortest path between two points [26]. In our work, A* is modified to compose multiple optimal paths so as to have the aggregation of distance, energy, and path duration determine the most efficient route. In a similar manner, we altered the traditional GA, and our HGA implementation leverages Ant Colony Optimization (ACO) to generate the initial population of promising paths, which are then further refined and optimized by the genetic algorithm. This hybrid approach harnesses the strengths of both ACO and GAs, enabling our framework to effectively identify energy-efficient routes in complex vehicular networks. Energy efficiency has become a paramount concern in vehicular transportation, prompting extensive research into optimizing PPAs to minimize vehicle energy consumption.

3. Methodology

In this section, an extensive overview of the essential components utilized within our framework is provided, accompanied by the theoretical underpinnings of the applied mechanisms. Our combined architecture is illustrated in Scheme 1. When formulating our approach, we aimed to create a comprehensive system that can be easily deployed while also examining the adaptability of an offline energy estimation model. This model is designed to align with the trajectories generated by the two PPAs we have incorporated.



Scheme 1. A high-level architecture of our framework.

As a framework, the user has the choice of selecting between the two PPAs. A* is capable of determining the shortest path based on distance in a very short computational duration. In contrast, the HGA searches for paths that satisfy three criteria, namely distance, time based on the minimum speed of the map, and a random weight that is assigned to each edge of the directed graph, defining the map topology for path diversification. We provide the option of employing both PPAs—one exclusively for a single simulation execution—in order to find the optimal paths and subsequently evaluate them through the utility of the energy estimation model. This indicates a modification of both PPAs to provide multiple paths instead of the singular, traditional best option, as there are cases where the second path might have a longer distance but less energy consumption, for example, due to a decreased number of stops. Hence, as it will be referenced later, the final selection stage checks the best path based on the aggregation of estimated energy and duration and the actual distance of the path. This will allow us to find the most efficient path solution and contribute to the design of sustainable autonomous vehicular networks. Our framework can also be viewed as a means and a basis for using Carla, towards path planning and energy estimation in general.

Initially, the PPA mechanisms are invoked, and the resultant solutions are processed to delineate links. As elucidated in [22], a link encompasses four phases representing distinct vehicle behaviors: acceleration, constant velocity, deceleration, and stationary. However, not all phases necessarily constitute a single link, as topological considerations influence their sequencing and inclusion. For instance, if a traffic light is detected at the end of the link, the deceleration and stationary phases are pertinent, whereas, in the absence of a stop, solely the acceleration and steady velocity phases are enacted.

Notably, the process of generating such links lacks explicit attention despite its significant influence on energy outcomes and vehicle behavior. One method to outline this constraint involves analyzing acceleration during the acceleration phase. Each road imposes a speed limit that vehicles must obey, serving as a means to regulate the maximum speed achievable while traversing the road. Achieving the prescribed velocity threshold necessitates computing acceleration for each unique phase, following an assessment of which phases contribute to the overall energy consumption calculation. Concentrating on the initial phase, the acceleration maneuver is imperative to ensure that the required

acceleration for the target speed is feasible within the phase's distance. As elaborated in the subsequent section, each phase's parameters—distance, acceleration, speed, average velocity, and estimated duration—are computed. If the necessary acceleration demands a distance greater than that of the phase, or in some instances, the link itself, then it becomes evident that acceleration cannot be executed, at least without violating the maximum speed limit. One solution to this challenge involves employing an iterative approach to identify the optimal acceleration and maximum speed achievable based on the link's distance. This approach offers the potential for vehicles to attain a substantial portion of the initial speed limit through reasonable acceleration while addressing the immediate complexity rising from decomposing the link into phases.

3.1. Energy Estimation Modeling

This subsection is greatly inspired by the work in [22]. The authors developed a detailed offline energy model estimator that is capable of processing the entirety of the desired path, with specified initial and final points, while taking into account the characteristics of the usable vehicle. Readers are encouraged to take a closer look at the EPA database in [27], which holds information for a plethora of vehicles, describing necessary metrics such as vehicle driving resistances. Since our work is an extension of this energy model tailored for the Carla simulator, it is important to showcase the energy model and the modifications we made.

This offline model is built upon the concept of links between points and decomposing that link into a multitude of phases describing a vehicle's behavior. More specifically, the topology of the map is transformed into a directed graph. The nodes are connected with weighted edges, where the cost is dependent on the employed PPA algorithm. To clarify this, we discuss in the next subsection the objective of each PPA, which affects the weights on the graph. Once this graph is constructed, the PPA takes place and returns to the best available paths with all the waypoints.

Waypoints is a concept of the Carla simulator that represents a 3D-directed point holding information regarding the location and orientation of that point in the lane containing it. We can also extract what that lane includes, meaning landmarks or, more specifically, stop signs, traffic lights, speed limits, junctions, elevation, etc. The energy model requires these as a way to define the distinct phases and calculation of unique physical values.

Each link can be decomposed into the following:

- (a) Acceleration phase;
- (b) Constant velocity phase;
- (c) Deceleration phase;
- (d) Standstill phase at the end of the link.

The road information stops at the ends of roads, or the type of road environment (highway, urban roads) determines which phases are enabled for each separate link. As a result, vehicle dynamics may not change between phases; nonetheless, for every phase, the average velocity, current speed, time of phase, distance traveled, and energy consumption are computed. When a phase is not executed, those elements are neutralized but still incorporated in the final phase energy consumption estimation. A simplistic example that showcases this process is when the vehicle is not obliged to stop at the end of the phase. Hence, phases (c) and (d) are excluded from execution. Incidentally, alters the velocity based on only the first two phases, but the final estimate results from the sum of all phases.

A battery electric vehicle (BEV) [28] model is used to measure the power consumed from the traction battery. Thus, battery power can be broken down into the power at the wheels and the consumed power from the auxiliary appliances where the traditional 12 V on-board supply and Heating Ventilation and Air Conditioning (HVAC) are considered. In addition, the model requires vehicle characteristics such as vehicle drag, weight, traction, and regenerative drive efficiency. All of these factors must be known before the initial model configuration. The obligatory power consumption formulas are identical to those in [22]. The power at the wheels can be established by leveraging the resistance from

acceleration, the average slope of the link to be traveled, and the average velocity. Below, we analyze how we calculate the individual components and then formulate the power for the wheels:

- *Average Velocity:* In our work, the maximum velocity of a link is indicated by the predefined speed limits of the map. When no speed limits are set, a maximum preset speed limit velocity is engaged, which, for simplicity, is the same for all the available maps. The average velocity is an immediate result of the initial and final phase speed; therefore, the measurement formula alters based on the phase type. On that note, we present how the initial v_{init} and final v_{end} velocities vary depending on the constraints of the link. In the case where a stop is mandatory, clearly,

$$(l) = 0 \quad (1)$$

with l denoting the current link. For every link,

$$v_{init}(l) = v_{end}(l - 1), \quad (2)$$

If no stop is set and

$$v_{max}(l) \leq v_{max}(l + 1), \quad (3)$$

then

$$v_{end} = v_{max}(l) \quad (4)$$

whereas, if the upper velocity limit of the current link is higher than the maximum speed of the next, then

$$v_{end} = v_{max}(l + 1), \quad (5)$$

Based on these velocities, the average speed of the link can be derived with ease depending on what phase the vehicle satisfies. Calculating the average velocity of each phase is

$$v_{avg^1} = v_{init}(l) \cdot 0.25 + v_{max}(l) \cdot 0.75, \text{ for phase a} \quad (6)$$

$$v_{avg^2} = v_{max}(l), \text{ for phase b} \quad (7)$$

$$v_{avg^3} = v_{end}(l) \cdot 0.25 + v_{max}(l) \cdot 0.75, \text{ for phase c} \quad (8)$$

$$v_{avg^4} = 0, \text{ for phase d} \quad (9)$$

- *Average Slope:* To calculate the average slope of the roadway in the link, we employ the trigonometric definition of the tangent functions in a right triangle, defined as the ratio of the opposite side (vertical change $\Delta z(l)$) to the adjacent side (horizontal distance, [29]).

$$d_{hor}(l) = \text{last link node} - \text{initial link node} \quad (10)$$

Thus, signified by ϑ , we utilize the formula

$$\vartheta(l) = \tan^{-1} \left(\frac{\Delta z(l)}{d_{hor}(l)} \right) \quad (11)$$

where d_{hor} is the horizontal distance given by the Euclidean formula as

$$d_{hor}(l) = \sqrt{\Delta x(l)^2 + \Delta(y)^2}, \quad (12)$$

and in both equations, x, y, z denote the 3D coordinates of the final and initial points.

- *Phase Acceleration:* Acceleration in the model can only be considered in phases (a) and (c) (negative acceleration); thus, in the rest of the phases, there is no variation in velocity, and the acceleration is set to 0. It is important to state that, in order to have realistic values, we created a table consisting of time values in seconds and the alteration of the velocity to describe as many distinct cases of acceleration as we could.

As it was a point of confusion for us when examining the model, we concluded that our approach was the more simplistic way to obtain the acceleration determined by the speed limits of the link. These were measurements based on the acceleration capabilities of the vehicle in the Carla maps for the various initial and final speed pairs. Thus, we can formulate the acceleration of the phases as

$$a_1(l) = \frac{\Delta v(l)}{t_{acc}(l)} \quad (13)$$

with Δv being the difference in velocities. Regarding phase (a),

$$\Delta v(l) = v_{max}(l) - v_{init}(l) \quad (14)$$

and accordingly for phase (c),

$$\Delta v(l) = v_{end}(l) - v_{max}(l) \quad (15)$$

In both cases, the $t_{acc}(l)$ is the time of acceleration or deceleration from the table. Subsequently, $t_{acc}(l)$ is also the estimated time of a phase's duration.

- **Wheel Force [22]:** As a contemplation of the above elements, we can obtain the wheel force through the vehicle driving coefficients A, B, C, and d, with the rotating mass coefficient resulting in

$$F_{wheels^k}(l) = A \cdot \cos(\vartheta(k)) + B \cdot v_{avg^k}(l) + C \cdot v_{avg^k}^2(l) + m \cdot d \cdot a_k(l) + m \cdot g \cdot \sin(\vartheta(k)) \quad (16)$$

where k signifies the phase the vehicle executes. Leveraging this formula, we can calculate the power of the wheels as

$$P_{wheels}(l) = \frac{F_{wheels^k}(l) \cdot v_{avg^k}(l)}{3.6} \quad (17)$$

After calculating the power of wheels $P_{wheels}(l)$, we combine the time of every phase in the link $dt_k(l)$ with the auxiliary power consumed and powertrain efficiencies. The phase time is a consequence of the distance traveled in a phase and the velocities and can be extracted as

- Phase (a): Time: with

$$\Delta v_1(l) = v_{max}(l) - v_{init}(l) \quad (18)$$

and distance

$$ds_1(k) = \frac{\Delta v_1^2(l)}{2 \cdot a_1(l)} \quad (19)$$

- Phase (b):

$$dt_2(l) = \frac{ds_2(l)}{v_{max}(l)} \quad (20)$$

and distance

$$ds_2(l) = D(l) - ds_1(l) - ds_3(l) \quad (21)$$

From this, we can understand that we first need to compute the distance of the next phase. That is a consequence of the selection of phases in the link to be carried out. If no further phases after this are executed, then the distance to travel is $D(l)$, the link's total distance, apart from the previous phase distance.

- Phase (c):

$$dt_3(l) = \frac{\Delta v_3(l)}{2 \cdot a_3(l)}, \quad (22)$$

where

$$\Delta v_3(l) = v_{max}(l) - v_{end}(l) \quad (23)$$

and distance

$$ds_3(l) = \frac{\Delta v_3^2(l)}{2 \cdot a_3(l)} \quad (24)$$

- Phase (d): since no movement is feasible, phase distance is set to zero, $ds_4 = 0$. However, the phase time is acquired by the type of stop if one is found. More specifically, we have created an average wait stoppage timetable that holds the duration of a stop due to a traffic light and a stop sign. These durations were captured by the simulator and during normal executions.

Assuming the vehicle will stop at every traffic light and stop sign it meets is a pessimistic way to interpret the vehicle's behavior. Overestimation is quite possible, if not a guarantee, for the estimation model performance. Nonetheless, having the vehicle take into account every possible stoppage in its path is a common practice for offline estimators. This ensures that if the energy obtained by the model is less than the real-world available vehicle energy, then the path is valid, and the vehicle can journey through it. As aforementioned, it is required to check if the link's distance suffices for acceleration to the maximum speed value or not. Specifically, we try to match the available travel distance with the maximum acceleration value that the vehicle can reach. This limitation means that we will reach the maximum feasible velocity in that time frame, and the link will end with a newly set maximum velocity. Depending on the upper limit of the next points, the vehicle will again begin from phase 1. Finally, we include the phase energy estimation in the form of [22]:

$$dE_k(l) = \left(\frac{P_{whk}(l) \cdot dt_k(l)}{n_d} + \frac{P_{aux}(l) \cdot dt_k(l)}{n_{aux}} \right) \cdot \frac{1}{3600} \quad (25)$$

In (1), the n_d is the average efficiency of the traction drive when consuming energy from the battery and n_{aux} is the average efficiency of energy recuperation from the traction drive back to the battery.

To obtain the total energy consumption for the path,

$$dE_{link}(l) = \sum_{k=1}^4 dE_k(l), \quad (26)$$

and it incidentally calculates the link's energy estimation and obtains the total path energy consumption estimation.

Finalizing the BEV energy estimation model required handling exceptions to the normal calculation. Such exceptions revolve around determining phases and stops, having short-distance links, and adjusting the acceleration. As previously mentioned, we have opted for an overestimation of the number of stops, including stop signs and traffic lights, thus considering the least favorable stop scenario, where the vehicle will stop at all possible stops. Highways are also a point of congestion as the vehicle will not stop when entering and immediately be set to the acceleration phase; hence, large acceleration is required to reach the maximum velocity, which usually hinders the performance of the model due to its deterministic acceleration approach.

3.1.1. Path Planning Algorithms

In this subsection, we will present the two distinct PPAs, A* and HGA, that we have incorporated into the framework. The utilization of such a mechanism is determined by the user, who can only use one of them in a single execution. Providing the utilization of these algorithms offers a nuanced approach to finding paths and solving optimization problems.

Traditionally, the A* leverages a heuristic that reflects a single objective, efficiently aiming towards a specific goal. In contrast, the GAs utilize an objective function ideal for complex and practical environments, which determines the best individual for the next generation. Objectives are suited to the path planning application they serve, with

predominant factors, for instance, route length, journey duration, as well as safety and smoothness [30], guiding the algorithms, thus influencing the search behavior. An appropriate objective function directs a more efficient search if it is less prone to exploring unnecessary paths.

3.1.2. Path-Planning Algorithm A*

The A* implementation is a feature inside the Carla simulator that employs a dedicated library to create and process the structure of dynamic and complex node networks. A* is a path-finding technique traversing an input graph with the ultimate goal of finding the shortest path [31] from an initial node to a target node. The graph requires a cost matrix to accompany it as weights. The practical advantage of this approach is the use of a heuristic to estimate the cost for each path generated, thus limiting the search towards the goal. Such a cost function can be expressed as

$$f(n) = g(n) + h(n), \quad (27)$$

with $g(n)$ being the cost of the path from initial to end node n , $h(n)$ being the heuristic operation, which is a flexible function constrained exclusively by never overestimating the actual cost, and $f(n)$ being the total estimated cost of a solution.

Since the implementation of the algorithm is already provided by the library *Networkx*, showcasing the practical aspect seems redundant. Therefore, we motivate the readers to explore the library and the A* altogether. What adjustments are commonly made regarding this PPA involve the tuning and objective of the heuristic. In our case, distance is the primary focus of minimization; hence, the Euclidean distance for 3D coordinates is an understandable approach with proven capabilities. In the case where no heuristic is provided, essentially, the A* is transformed into the accepted Dijkstra's algorithm, setting a zero heuristic. Furthermore, we adjusted the A* approach to extract multiple solutions instead of the best one to determine the best path based not only on distance but energy consumption and estimated duration as well.

3.1.3. Path-Planning Algorithm HGA

An improvement over the traditional GAs in terms of decreasing the computation time is required to reach an acceptable solution. For real-world applications, the most time-consuming component usually is the search speed, an immediate effect of the objective and fitness function [11]. GAs demonstrate substantial improvements regarding convergence speed when combined with local search methods or hybrid optimization, capturing the best attributes of the components comprising it [32]. Thus, the convergence speed is a critical factor, synonymous with the performance of the HGA, and is subject to the increasing population. As showcased in [33], two hybrid approaches were addressed when increasing the fraction of the population and results dictate the expected increase in convergence speed. The authors note that the increase is not linear. Furthermore, in [34], the authors outline that a larger and more extensive search space increases the diversity of desired results, which ultimately assists the HGA by enhancing its exploration abilities and helping the algorithm avoid local optima. Finally, in [18], the author admitted that a sufficiently large number of constraints as a consequence of the increased search space is not an obstacle for the HGA because of its hybrid nature that combines global and local search strategies to find feasible and optimal results in a more efficient manner.

Our consideration for the HGA was to use a meta-heuristic optimization mechanism to initialize the population, which is then processed by a fundamental GA. An HGA aims to combine the beneficiary attributes of different algorithms to surpass their individual restrictions and weaknesses. A widely known and accepted meta-heuristic approach for path-finding combinatorial problems is the Ant Colony Optimization (ACO) [35]. We deploy our version of ACO to construct the starting population by exploiting the ACO's ability to generate high-quality initial solutions that guide the GA towards promising regions when exploring the solution space. The GA has the capability to refine, over

successive generations, the solution through its evolutionary processes. The ACO utilized for the initialization of the population can become computationally expensive as the problem size increases due to the maintenance of pheromone trails which ACO depends on [36]. HGA is a more scalable option for larger problems due to its flexible population size, and in our case, we kept the population size to be the direct result of the ACO solutions with static lower and upper bounds so as to satisfy constraints regarding applicable paths. Consequently, our hybrid approach distinguishes itself by the generation of the initial population phase, where the classical random or often heuristic-based initialization is ignored or supplemented with a more structured and defined method. This strategic synergy not only capitalizes on the exploratory and intensification capabilities of ACO but also harnesses the diversity and generational evolution strengths of GAs, creating a more comprehensive optimization framework.

Initialization of Population: As stated in [35,36], ACO is inspired by the foraging behavior of real ants, finding paths from their colony to food sources. When searching for food, ants will explore a region in a random manner, centering around their nest. While on the move, ants release a pheromone on the ground, and the intensity of that substance informs the ants to either follow the trail or not. This iterative process leads to the discovery of the more appropriate paths, based on the objective function, as they become reinforced with pheromones at a higher rate. The pheromone model defines how pheromones are deposited and how they evaporate over time. This determines the selection of nodes also affected by the visibility (or attractiveness) of each path, which is inversely related to the cost metrics depending on the use case. Such a process, conversely, can be translated into a calculation of edge selection probability.

For a constructed directed and weighted graph, we denote each edge as e_{ij} , from node i to node j . During selection, j belongs in N , the unvisited nodes. We set a pheromone, τ_{ij} and if the edge e_{ij} does not have a pheromone value directly, we leverage the pheromone value from e_{ji} , ensuring symmetry. Visibility of the edge, v_{ij} , is calculated as

$$v_{ij} = \left(\frac{\alpha}{w_{ij}} \right) + \left(\frac{\beta}{t_{ij}} \right) + \left(\frac{\gamma}{ds_{ij}} \right), \quad (28)$$

where in the formula, α, β, γ are parameters that control the relative importance of pheromone strength, the inverse of weight, the inverse of time, and the inverse of edge distance. Thus, our objective function is set to use the time of travel for the distance of the edge along with an assigned weight to distinguish edges that might have the exact same distance and time values. The next node is randomly selected according to the probability P_{ij} proportional to

$$P_{ij} \propto (\tau_{ij})^\alpha \times (v_{ij})^\beta, \quad (29)$$

with α and β controlling the influence of pheromones and visibility. Then, the probability of each feasible edge is normalized by dividing the sum of probabilities for all feasible edges to ensure that the sum of all probabilities equals 1:

$$P_{ij} = \frac{(\tau_{ij})^\alpha \times (v_{ij})^\beta}{\sum_{k \in N} (\tau_{ik})^\alpha \times (v_{ik})^\beta}, \quad (30)$$

If the total probability

$$\sum_{i,j \in N} P_{ij} \leq 0 \quad (31)$$

indicating an issue with the probability calculation, which can be the result of no unvisited nodes having a feasible path, then the algorithm resets by selecting the source node as the next node and clears the visited nodes set to restart the path. Resetting the path requires the ant to reset into the start from the source node again; hence, a feasible path could not be established from the current position, and the visited nodes are also cleared for a fresh start.

Pheromone evaporation and deposition of new pheromones are the remaining steps for the ant based on recent paths. For every edge e_{ij} in the pheromone matrix,

$$\tau_{ij} \leftarrow \tau_{ij} \times (1 - \rho) \quad (32)$$

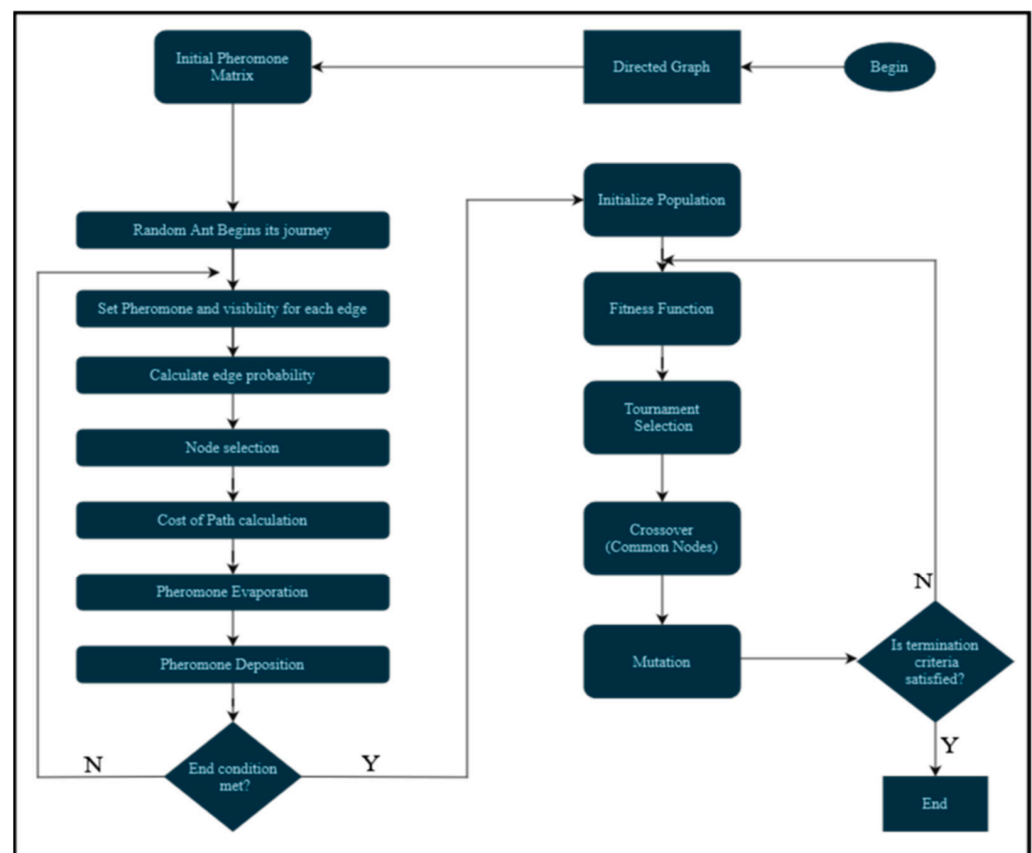
with ρ being the evaporation rate, a parameter of the algorithm that controls the rate at which pheromone evaporates over time. This ensures that the algorithm does not converge prematurely and explores new paths. For each path traversed by an ant, an additional pheromone is deposited on the edges of that path based on the quality of the path (inversely related to the cost). Next, given a path P consisting of nodes $n_1, n_2, \dots, n_{\xi}$, with $\xi \in A$ traversed in order (A is the set of nodes for P), the cost of the path, denoted by $C(P)$, is calculated based on the previously mentioned edge attributes:

$$C(P) = \sum_{i,j \in A} t_{ij} + ds_{ij} + w_{ij}, i \neq j \quad (33)$$

As a result, the pheromone deposition is depicted as

$$\tau_{ij} \leftarrow \tau_{ij} + \frac{\Delta\tau}{C(P)} \quad (34)$$

In this last formula for the ACO, $\Delta\tau$ is the amount of pheromone deposited, which is constant. This update process incorporates both the fading of older information (evaporation) and the integration of new information (deposition), reflecting the collective learning process of the ant colony. This balance is crucial for the successful discovery of the best paths. The complete architecture of the HGA is demonstrated in Scheme 2.



Scheme 2. HGA fundamental structure.

Fitness Function: For each edge in the path, we extract the three attributes and assign a scale value to each one as $\lambda_{time}, \mu_{dist}, v_{weight}$ to establish the hierarchy of importance to the attributes. Here, in order to be able to compare it to the A^* , we decided that distance ought to mainly affect the fitness score. We sum each attribute value for all applicable edges in the path, and we assign a large enough penalty for edges that do not belong to the graph. We then normalize the summed attributes based on upper limits set from multiple observations during execution as $max_{time}, max_{dist}, max_{weight}$. The fitness score can be derived by

$$F(P_x) = \left(\lambda_{time} \cdot \left(\frac{\sum_{ij \in A} t_{ij}}{max_{time}} \right) + \mu_{dist} \cdot \left(\frac{\sum_{ij \in A} ds_{ij}}{max_{dist}} \right) + v_{(weight)} \cdot \left(\frac{\sum_{ij \in A} w_{ij}}{max_{weight}} \right) \right), \quad (35)$$

where the formula transforms the objective into a maximization problem, and we are searching for the lowest value of this aggregation.

Selection: *Tournament selection* involves a random subset selection derived from the population (representing the tournament) and then choosing the best individual examined by the fitness score from this subset and transporting it to the next generation. More accurately, given a population G , choose, at random, m individuals to define a tournament of size m . This iterative process starts by obtaining m individuals randomly from G into a tournament group T . After evaluating the fitness value for every individual with $F(T_i)$ a fitness score of the i -th individual, we can elect the one with the best fitness score to be a part of the next generation's population:

$$I_{selected} = \operatorname{argmin}_{i \in T} F(T_i) \quad (36)$$

Crossover: Substantially, crossover is the operation of combining parts of two selected "parent" paths to create an "offspring" path, potentially inheriting beneficial traits from both parents. Given two parent paths P_1, P_2 in our model, we identify the common nodes between the two, forming a set of potential crossover points, Q . Then, we randomly select a crossover point if the data structure Q is not empty. Crossover point $q \in Q$, we can construct the offspring path by concatenating the subpath from P_1 up to q with the subpath P_2 after q , validating the path once the process is finished. As a result, the offspring path O can be represented:

$$O = [P_1[1], \dots, P_1[q], P_2[q+1], \dots, P_2[n], P_1[n]] \quad (37)$$

where $P_1[1] \wedge P_1[n]$ are the initial and final nodes.

Mutation: Mutation operation introduces random variability through alterations in the individual's path, providing the opportunity to explore past unseen areas of the search space. With a probability equal to the mutation rate, we can select two nodes within the individual I , without considering the required start and endpoints. Once again, we have to ensure the generated path is valid for the nodes that exist in the graph. Mathematically, if the mutation occurs, the altered individual I' after swapping nodes at indices,

$$\delta, \epsilon: I' = [I[1], \dots, I[\delta], \dots, I[\epsilon], \dots, I[n]] \quad (38)$$

with a subsequent validation step to ensure all nodes in I' are valid within the graph context.

3.2. Modeling the Framework

In this subsection, we discuss in detail the path processing method we utilized to mitigate the number of nodes needed and transform the path into a list of links in order to be accessible to the energy estimation model. We also present interfaces that we utilize to set the depot and target destination in a dynamic manner and how we evaluate the best path after each separate execution.

3.2.1. Douglas–Peucker Model for Link Creation

The Douglas–Peucker algorithm is also known as an iterative end-point fit algorithm with the goal of smoothing polylines (linear line segments composing lines) [37]. This process ultimately reduces the number of points, but the newly defined curve should preserve the rough shape of the original curve. The level of coarsening is a consequence of the parameter epsilon, or ϵ , that limits the maximum distance between the original points and the simplified curve [38]. In our case, we implemented a variation of this algorithm, adjusted for path simplification based on curvature.

More specifically, our model starts with the curvature calculation. Given three way-points p_1, p_2 , and p_3 with coordinates (x_1, y_1, z_1) , (x_2, y_2, z_2) , and (x_3, y_3, z_3) , correspondingly. We define vectors u_1, u_2 to hold the positional difference among $p_1 \rightarrow p_2$ and $p_2 \rightarrow p_3$, respectively. Next, we compute the dot product of the two vectors as

$$\sigma = u_1 \cdot u_2 \quad (39)$$

Thus, we can combine the vectors and σ to calculate the angle φ between the two vectors leveraging the fundamental cosine angle formula:

$$\cos(\varphi) = \frac{\sigma}{\|u_1\| \|u_2\|} \quad (40)$$

and find the angle as

$$\varphi = \arccos(\cos(\varphi)) \quad (41)$$

This angle denotes the curvature at p_2 . Also, we define a curvature angle threshold as φ_{thr} to categorize segments as curved and add the point to the list of key points if the measured curvature is greater than the threshold. Thus, the main step of our model is to iterate through the path's points and calculate the curvature at each point to then obtain the key points of curvature. Our variation to the traditional approach aspires to simplify a path by recursively reducing points based on curvature and segment classification.

At this point, the key points have been observed and we can classify segments of the path as “curved” or “straight” through them. Here, the recursive prospect of the model is presented, as for every segment, we apply the Douglas–Peucker simplification based on the appropriate ϵ value, now divided into ϵ_{curved} and $\epsilon_{straight}$ for curved and straight segments, respectively. Furthermore, a focal point of our model is the simplification process for a segment S where there is a perpendicular distance from each point in S to the line formed by the first and last points of it. Now, we are capable of determining the point with the maximum distance ds_{max} from the line, and if

$$ds_{max} > \epsilon \quad (42)$$

recursively apply the simplification process to the sub-segments created by splitting S at the ds_{max} point. Otherwise, the segment is adequately simplified, and only the first and last components are necessary. The process continues until all segments based on their classification are simplified. The final output is a simplified path that approximates the original path within the given tolerance levels. Together, these individual components form a path simplification algorithm that takes into account the path's curvature, preserving critical geometric features while reducing the number of points. It is a suitable approach for paths where the shape, rather than just the path's endpoint, is important, such as in autonomous vehicle navigation or map features.

In Figure 1, the map and the path provided by the PPA algorithm are showcased. The density of nodes representing roads is large, as it should be, during execution, as each waypoint describes if the vehicle should change lanes, turn or go straight, etc. However, once again, the links to be created should be above a minimum distance threshold to allow phases to be executed. Thus, we need to find and preserve the key nodes to create applicable links. Initially, we apply a filter to ensure the minimum length of the edge is

met between two nodes in the path. This approach removes a multitude of nodes from the path. The filtered waypoints are grouped by road and are fed into the Douglas–Peucker variation. The process is captured in Figure 2. A significant number of nodes have once again been disposed of, and links now match the roads of the network.

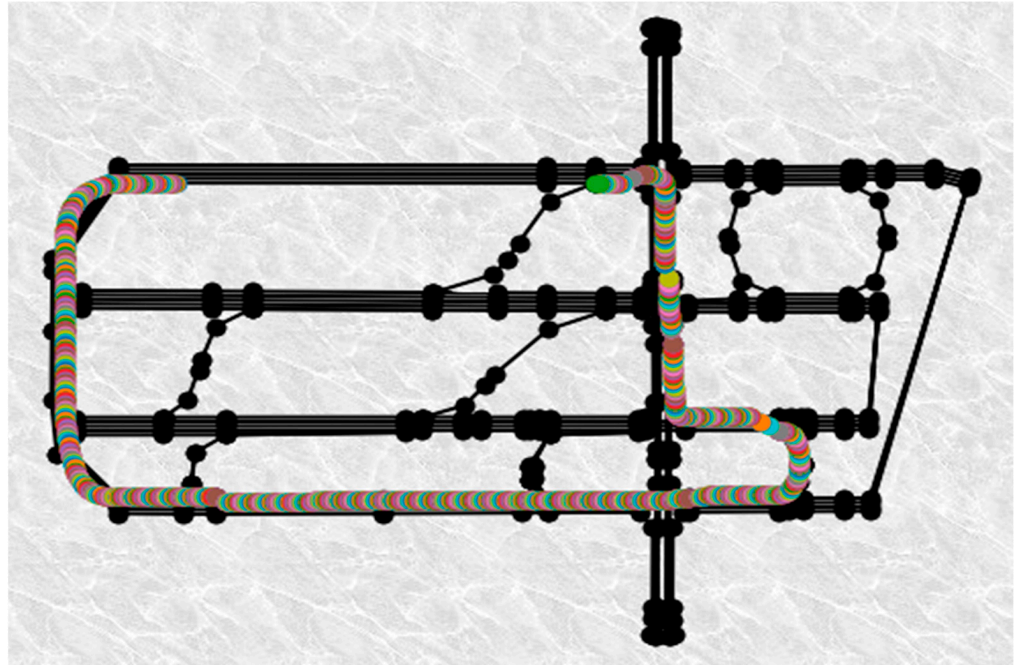


Figure 1. Depiction of the entire road network for Town 06 from Carla maps, showcasing the initial path returned by the A* PPA. The colored nodes denote the path and all the waypoints that the car must pass through.

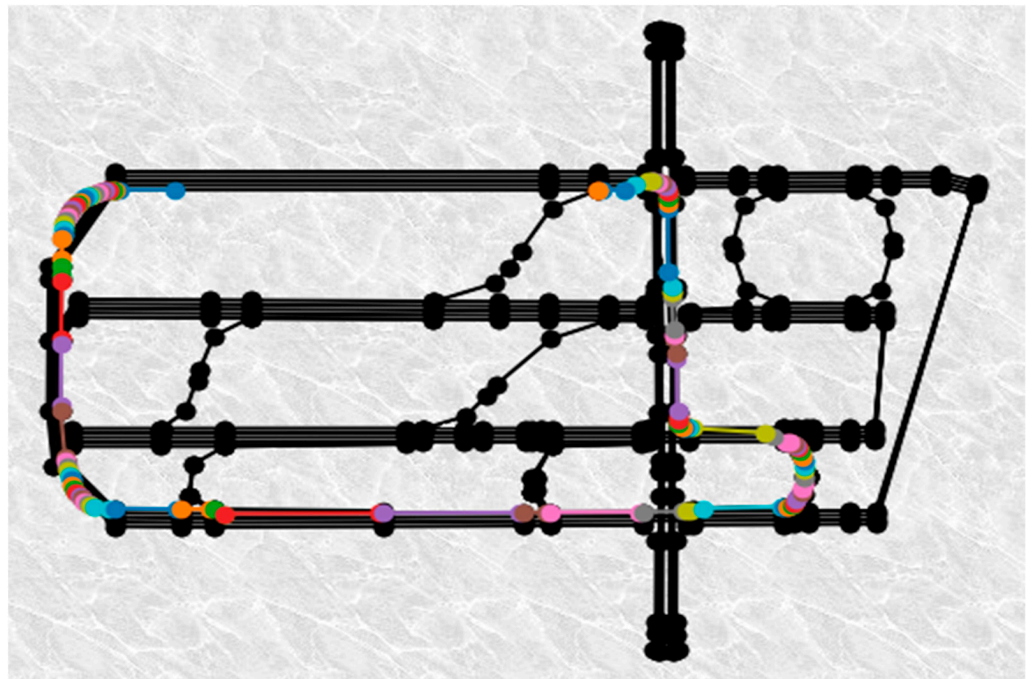


Figure 2. Depiction of the entire road showcasing the filtered path returned by the Douglas–Peucker variation.

The final Figure 3 captures the finalized form of the path after the last simplification in the curved parts of the route, which, after the first filter stage, holds the most nodes. The entirety of this procedure makes the creation of applicable links possible and also allows the division into phases without compromising the solution of the PPA.

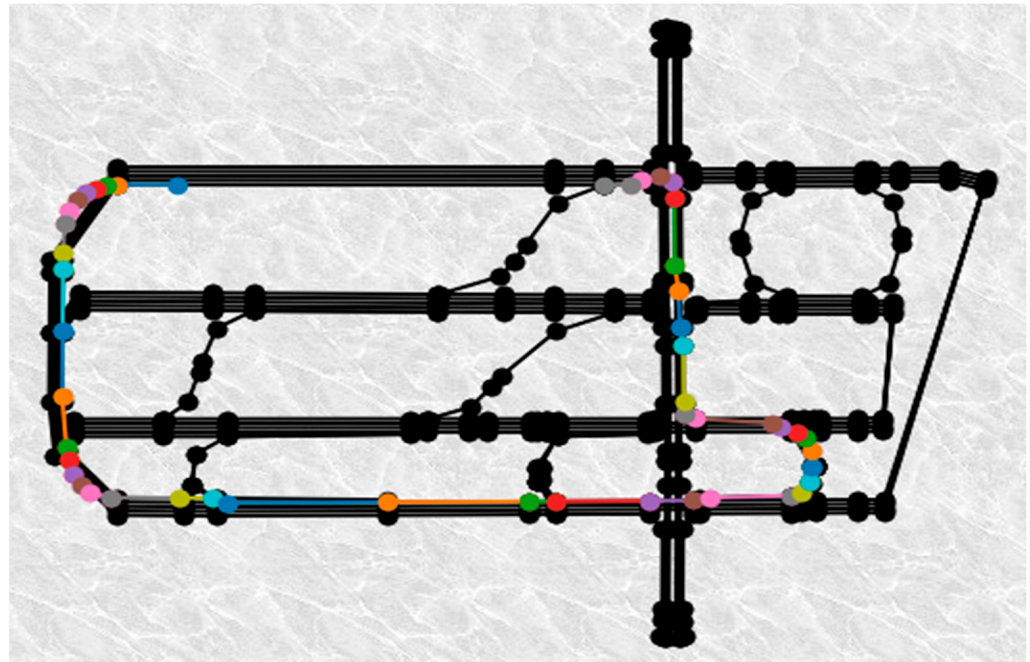


Figure 3. Last simplification of the path. This will be the final path that the estimation model will consider.

3.2.2. Optimal Path Evaluation

For the validity and comparison of the two distinct path-planning approaches, we considered a multi-criteria evaluation scheme which focuses on primary objectives and secondary metrics. The former consists of path Optimality, where we measure the solution's alignment with the main optimization objective that each algorithm aims for. Energy efficiency is also a primary performance indicator, as it provides a common ground to compare the outcome based on real-world applicability. The latter is composed of metrics such as smoothness, Node Population, computation time, flexibility, and scalability.

- A smoother path is generally preferable for practical applications as it ensures passenger safety and comfort.
- The number of nodes reflects the complexity of the path. Fewer nodes might indicate a more straightforward and potentially easier-to-navigate route.
- Especially in near real-time applications, computation time is essential for proving the practicality and scalability factor of the algorithm.
- Flexibility is defined, in this case, as the ability to adjust to distinct optimization goals, or in other words, how difficult is the modification of the algorithm in order to prioritize another objective instead of the traditional goal (e.g., distance).
- Scalability evaluates the performance when the problem space increases, as is the case with the deployment of a larger map.

Lastly, we ought to mention that the catalyst distinguishing the optimal path is a culmination of the different components that justify the selection of a path. Our methodology is an optimization algorithm that evaluates multiple potential routes (P) based on a linear combination of estimated energy cost (E), travel distance, and estimated travel time (T). After initial experimentations, we observed an immediate effect that the number of stops had on the overall path quality. Thus, we included an additional factor S , denoting the total number of stops. We additionally normalized each metric to a common scale of before

applying weights to address the importance of each attribute. Thus, we can assume the following formula for the overall score, C , of the path

$$C_p = w_E \frac{E_p}{\max(E)} + w_D \frac{D_p}{\max(D)} + w_T \frac{T_p}{\max(T)} + w_S \frac{S_p}{\max(S)}, \quad (43)$$

We ensure that no critical metric is missing for any path, as well as that the path itself is applicable to the road topology; otherwise, an empty path is issued. With this, we can select the route that minimizes the aggregate cost, ensuring robust decision-making supported by the secondary criteria that only act as a safety valve to demonstrate the best and worst characteristics of the optimal path provided. This also enhances the importance of the essential path attributes and provides a logical comparison and evaluation of both the PPAs.

3.2.3. Dynamic Point Allocation

During the conceptualization of our work, we wanted to create a comprehensive framework to combine the capabilities of the Carla simulator and expand the routing mechanisms used while honing an energy estimator. Setting aside the PPA, the energy model, and the simplification techniques for the solution, we aspired to develop a dynamic framework which we can build upon.

More specifically, we fully utilize major Carla capabilities and components it offers with no intervention. Our work focuses on expanding those components. First and foremost, the configuration of the source and target locations to generate the appropriate path is required to be dynamic and accessible. By developing the interface in Figure 4, we set all the available spawn points to position the vehicle in a correct manner on the map, and we ensure that the target waypoint is also feasible. The user is then able to select every point on the image, and the program will find the closest spawn waypoint from the map topology. This feature ensures the normal behavior of the PPA and the vehicle's control during its journey.

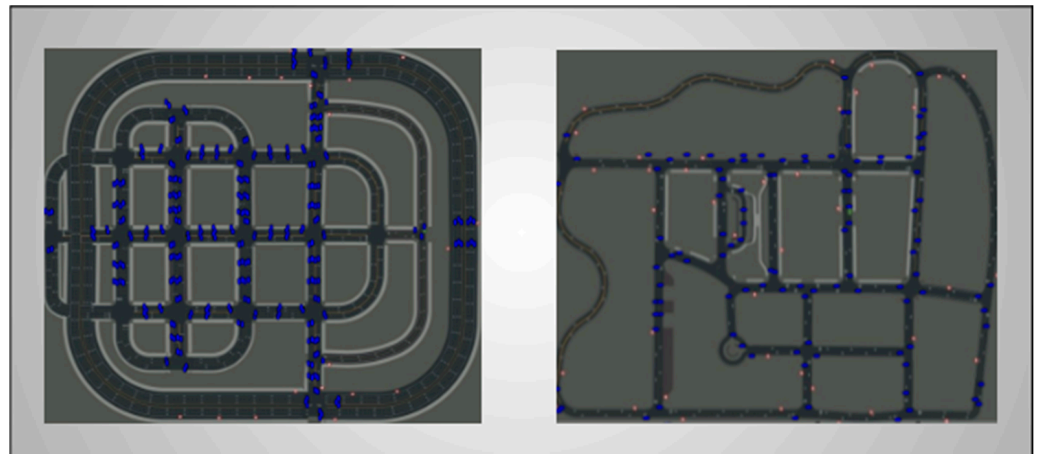


Figure 4. Demonstration of how the interface shows the applicable spawn waypoints to the users and demands the selection of depot and destination. Here, Town 05 and Town 07 are showcased.

However, we state that some waypoints are not perfectly aligned with the map, as illustrated in the map pictures, due to cropping and resizing of the image in order to use it with the corresponding tool for the interface. This means that sometimes, the selection between two different points on the map close to each other may result in the same waypoint selection. This is also a consequence of not having a reference point for each map. The user is capable of continuously altering their choice of points, but they are required to select both the starting point and the destination. Otherwise, the program is prohibited from execution, and the simulation stops. We admit that all the available maps are included

for the interface, except for Town 08 and Town 09, which are not publicly available, and Town 11 and Town 12, which are quite large in scale and need a different approach in order to be functional.

Additionally, we endeavor to adapt our framework to accommodate all available Carla. The simulator permits full map reconstruction during the initial setup phase, provided the server is not operational (running). Nonetheless, the enlarged map dimensions substantially elevate the complexity of the two PPAs, leading to extended computational durations, particularly concerning the HGA. Consequently, the graph's configuration varies depending on the map. Thus, with larger-scaled maps, both PPAs become more computationally complex due to their dependence on the generated graphs.

3.2.4. Vehicle Modulation

Within our framework, users enjoy the flexibility to input their vehicle details from the existing Carla blueprint library or by constructing them themselves if necessary. While our energy model primarily relies on a few key vehicle attributes, such as the rolling distance coefficient and aerodynamic drag profile, we offer a comprehensive parameterization option. In the simulator, the vehicle is treated as an entity with specified physical controls (e.g., gearbox, maximum RPM), characteristics (e.g., doors, lights, wheels), and access to various parameters like steering, speed, acceleration, and brake controls. Additionally, the simulator allows for the modification of the vehicle's PID controller if desired. This ensures that the vehicle's dynamics align with Carla controllers while incorporating the energy model. We utilize JSON files to store these details for ease of management. Our framework is designed to deserialize these files into understandable elements, which can be further modified to include additional information as needed. Within our primary code snippet, users can generate the required JSON file, providing all desired characteristics. This effort aims to incorporate the maximum amount of data into a default configuration, enhancing usability and comprehension. Furthermore, we include acceleration values used in the energy estimator, derived from the initial and final velocities, within the JSON file. While the capture process for each velocity combination can be laborious, users are encouraged to utilize the provided prototype and adjust the main values accordingly, particularly if the difference in vehicle performance is not significantly extreme.

4. Simulation Results

In this section, we will present, in detail, the numerical results of our framework, which mainly revolve around the energy estimator and the PPAs. A single execution is defined as the selection of the depot and destination, then the processing of the vehicle's characteristics and execution of exclusively one PPA. Each PPA constructs multiple paths and feeds them to the energy estimator. From there, we calculate the total energy consumption for each one of the paths, and by comparing the total energy consumption, estimated time, and estimated distance, we determine the most optimal path for the vehicle.

To make the results more robust and scalable, we opted to compare the performance of the framework for all the available maps. The similarity in topology and size between a few maps results in some closely defined conclusions, but it also highlights the immediate difference once the map significantly changes. The examination was conducted on a single machine, occupied by the latest version of the Carla simulator [5], leveraging modules already existing in it and our own work. We consistently utilized a virtual version of the Audi E-Tron included in the simulator, and we specified the model of the car to have some of the characteristics of the Audi E-Tron 55 Quattro [39] to use as a reference for the results. We executed each path twice to validate both PPAs and each map for a total of four routes to have diversified results. Table 2 presents the fundamental constant values employed in every execution of the framework.

Table 2. Simulation parameters.

Parameters	Values	Description
$m_{vehicle}$	kg	Mass of the vehicle (Unladen)
A	N	Vehicle driving resistance coefficients
B	$\frac{N}{(\frac{km}{h})}$	
C	$\frac{N}{(\frac{km}{h})^2}$	
P_{aux}	W	
$max_{population}$		Average power consumption of auxiliary drives
α, β, γ		The maximum number of Initial routes produced by the ACO
$\lambda_{time}, \mu_{dist}, v_{weight}$		Coefficients determining the objective function of ACO
traffic_lights_stop_time	s	Coefficient determining the objective function of GA
acc_time	-	Array of stoppage times when vehicle meets a traffic light
ϑ	radians	Set of initial and final velocities indicating the respective acceleration
$\epsilon_{curved}, \epsilon_{straight}$	-	Average slope of a link
		Coefficients separating curved and straight segments in the Douglas–Peucker approach

Each map possesses distinct characteristics, gradually transitioning from simplistic small towns to extensive highways with multiple lanes. Consequently, we encompass the essential environments for electric vehicles, albeit excluding mountainous areas or regions with significant altitude variations, though more approachable mountainous areas are considered. Moreover, the scale of the maps is relatively modest, prompting us to evaluate the framework for distances up to 4 km. Theoretically, we can test larger distances; for example, both PPAs often compute paths exceeding this upper limit by a considerable margin, sometimes reaching up to 20 km. However, this may involve traversing all available nodes of the map and occasionally inadvertently looping back or revisiting the same area from different nodes.

The vehicle's motion is entirely simulated, including the potential generation of traffic, which impacts the performance during the journey. However, traffic does not factor into the energy estimation model, nor does weather. A more comprehensive approach would involve incorporating air resistance and testing the model across different seasons. Nevertheless, according to the simulator, dynamically changing weather conditions only affect the visual environment; they do not influence vehicle performance. Additionally, road conditions are not considered, as each map features perfectly structured roads without potholes or irregularities. Concluding the simulation configuration, there are instances where both PPAs may struggle to find a feasible path between the specified initial depot and destination.

The simulation results focus on the aforementioned primary and secondary objectives to compare the performance of the two distinct PPAs and check the validity of our implementation of the energy offline estimation model. Taking a closer look at Figure 5, we outline a holistic overview of the energy estimation for all the paths of the various maps utilizing both PPAs. This figure shows a proportional relationship between the estimated consumption and the distance that the path covers. As evident by the formulas used to estimate the energy consumption, the energy increases alongside the distance growth. It is clear that the choice of PPA can significantly affect the energy consumption estimates, and their performance can vary greatly depending on the environmental complexity and the characteristics.

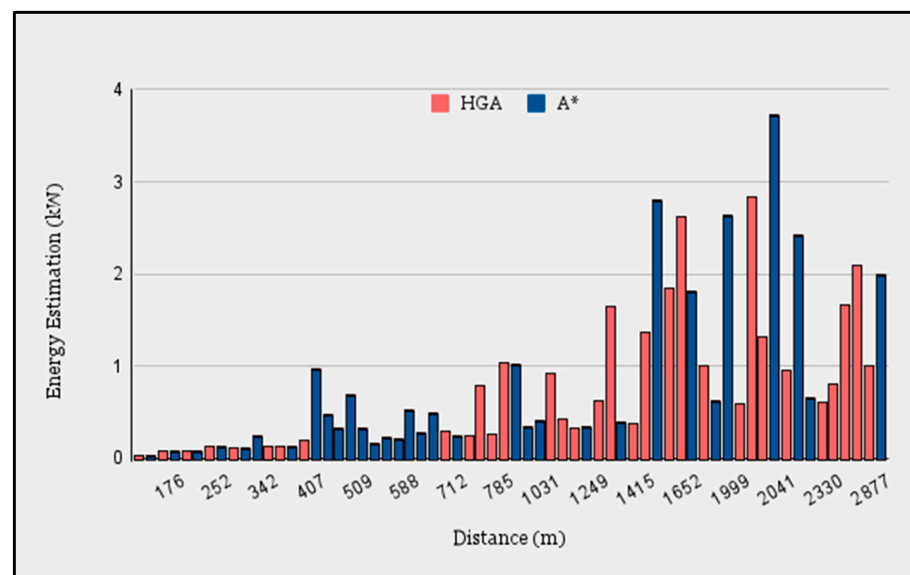


Figure 5. Energy estimation based on the offline model compared to the distance. This is a combination of all the maps available.

Since all maps are part of the diagram, we can deduce results on the impact of the environment. More specifically, we can focus on mountainous areas and junctions and large maps with highways of multiple lanes, as these maps have the biggest discrepancy in results. The former seem to be more challenging for the A* algorithm, leading to higher energy estimates, as such map features along with the topology may cause the A* to compute paths that involve frequent stops, starts, and elevation changes, all of which increase energy consumption. We focus on the A* as it produces the more radical behavior, but this concept applies to both PPAs. As for the larger maps, their more straightforward environments allow for more efficient path planning, closer to real vehicle efficiency. Based on the efficiency profile in [39], the performance of the Audi E-Tron 55 quattro is $237 \frac{\text{Wh}}{\text{km}}$. The best performing map for our model is Town 06, with multiple lanes, long roads, highway entrances and exits, and a Michigan Left. A closer-detailed look at the results regarding Town 06 can be seen in Table 3 for both PPAs. We should note that for this map, the average distance for A* is 1661.5 m with an energy estimation value of 512 Wh, and correspondingly, for the HGA, it is 1700.25 m with an energy estimation of 483.75. That means that the efficiency per kilometer is, for A*, $308.155 \frac{\text{Wh}}{\text{km}}$ and is, for HGA, $284.516 \frac{\text{Wh}}{\text{km}}$, which are close to the real performance of the vehicle.

Table 3. Overview of all the results for the distinct routes for Town 06.

PPAs' Routes	Route 1	Route 2	Route 3	Route 4
Energy Consumption (A*) (Wh)	629	396	354	669
Energy Consumption (HGA) (Wh)	598	383	338	616
Distance (A*) (m)	1966	1332	1200	2148
Distance (HGA) (m)	1999	1387	1183	2232
Duration (A*) (s)	412	251	215	431
Duration (HGA) (s)	394	245	206	404
Smoothness (A*)	0.544149851	1.17618481	0.891987664	1.18616680
Smoothness (HGA)	0.482322979	1.05136936	0.883766188	1.16540700
Total number of nodes (A*)	983	669	590	1000
Total number of nodes (HGA)	949	626	542	992

Table 3. Cont.

PPAs' Routes	Route 1	Route 2	Route 3	Route 4
Total number of stops (A*)	6	4	3	7
Total number of stops (HGA)	6	4	3	6

Additionally, Table 3 reveals the connection between the energy estimation and individual components, such as the total number of nodes and total number of stops. These relations are proportional, wherein the greater the number of nodes, the more energy is needed, which is a similar rule for the total number of stops. Estimated duration is another element that is impacted in the same way, proportional to the total distance of the path. We can also obtain a first glimpse of the importance of a smooth path and how it affects energy estimation. A smoother path is always preferable as it is safer and often more comfortable. The lower the smoothness value, the lower the energy consumption will be, and we can deduce that the HGA offers more preferable routes to the user with slightly higher distances.

An interesting validation approach is to examine how the smoothness of the path affects the estimated duration for each algorithm. Figure 6 demonstrates the smoothness values for all executions, divided once again for the PPAs. It outlines that HGA is a fundamental robustness choice for maintaining and improving the quality of path smoothness as paths become longer, whereas for A*, adjustments or optimization are required to enhance its performance for longer or smoother paths, possibly by incorporating additional criteria focused on smoothness. A significant portion of A* points are clustered above a smoothness of 0.8 and under a duration of 300 s. This suggests that A* is generally capable of quickly computing paths with decent smoothness but might struggle to maintain or improve smoothness in longer duration paths. Another way to interpret this is to consider that the A* might generate paths that are either too rough or too smooth relative to their length, suggesting potential inefficiencies or inconsistencies in path quality over varying durations. On the other hand, HGA shows a proportional, logarithmic-like increase in smoothness with duration. This suggests that HGA is more effective at increasing path smoothness in a gradual and predictable manner as path duration increases, indicating a better overall quality management of path quality over time. As a reference point, the average smoothness factor for A* is calculated at approximately 2, and for the HGA, it is around 1.80 for the entirety of the experiment, even though HGA holds the maximum smoothness values.

Setting our interest in the comparison of the energy estimation between the maps, we present Figure 7. This figure consists of two subplots: the left plot showcases the energy estimation performance of the A* algorithm, while the right represents the performance of the HGA. Both subplots include all tested maps and depict the distribution of energy consumption for each algorithm. From the figure, two overarching observations can be made: town variability and consistency across PPAs. Firstly, it is apparent that environmental and map-specific characteristics significantly impact the algorithm's efficiency, a notion previously established. Additionally, several maps exhibit similar energy consumption patterns between A* and HGA, indicating that underlying map features influence energy consumption regardless of the algorithm employed.

In-depth analysis reveals that Towns 01, 02, and 10 exhibit remarkably similar energy distribution patterns for the A* algorithm, with a notably low upper limit at 300 Wh. This suggests highly efficient path planning in less complex environments. Correspondingly, the utilization of HGA shows a comparable trend for Town 02 and 10; however, there is a slight deviation for the Town 01 map, with an expanded distribution and higher maximum energy consumption. In contrast, Town 03 displays a wider range, with outliers reaching approximately 1000 Wh, indicating more challenging navigation conditions. The HGA also demonstrates a broader distribution, albeit with a slightly higher value range and variability indicated by the position of the box on the line. The most significant differentiation is

observed in Town 04, presenting the widest energy distribution from 500 Wh to 3700 Wh, with the central box encompassing this range. This pattern is mirrored in the case of HGA, with a maximum limit of 2800 Wh.

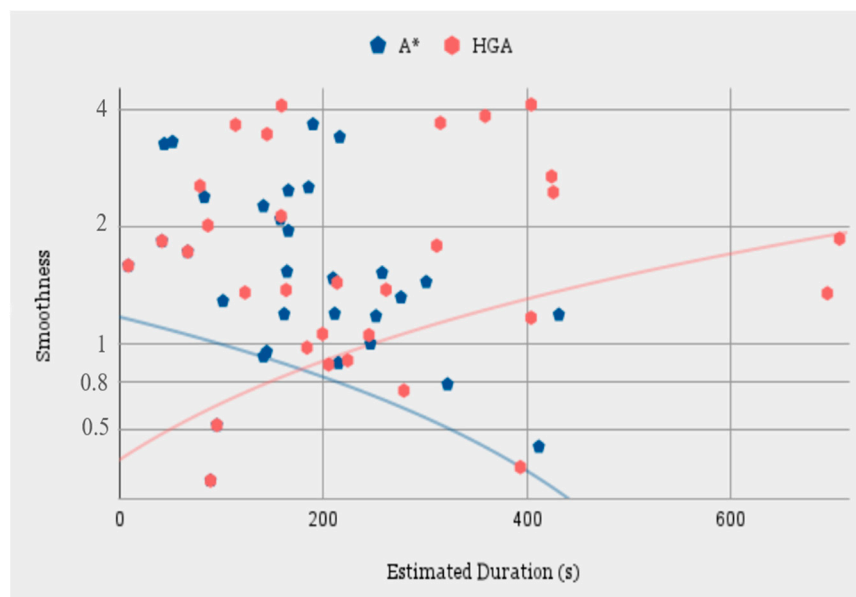


Figure 6. Complete smoothness representation for the entirety of the experiment and the impact towards the estimated duration for both PPAs.

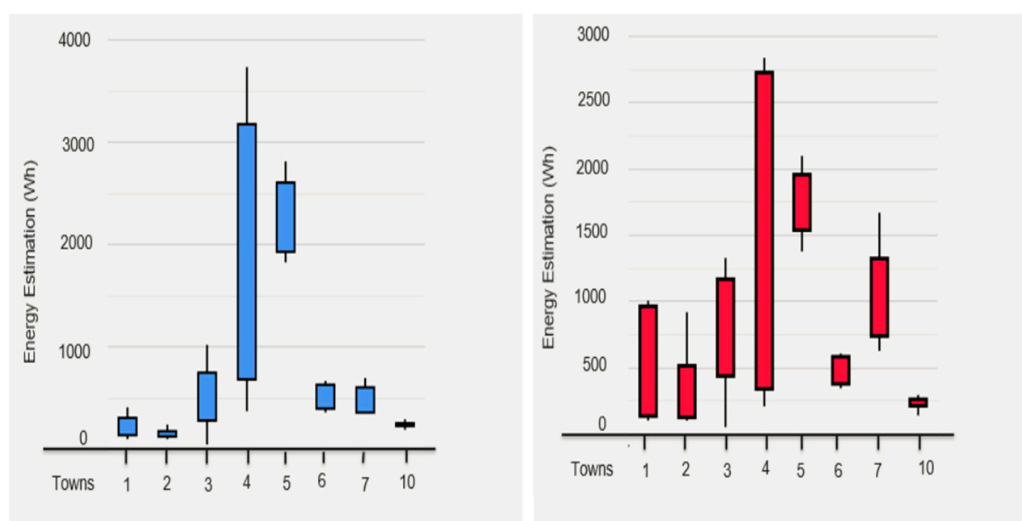


Figure 7. Two plots showcasing the energy distribution for all maps. The left plot represents the energy behavior as a consequence of A*, and correspondingly, the right is with the HGA in use.

Regarding the rest of the maps, for Town 05, the A* demonstrates a smaller grouping range from approximately 2000 to 2600 Wh, indicating consistent yet high energy consumption. This suggests that multiple junctions create a more complex grid, disrupting the operation of the PPA. Conversely, for the HGA, the energy follows a lower value distribution with a similar size, representing an improvement over the A* but still reflecting high energy usage due to specific environmental features. In the cases of Town 06 and Town 07, the performance of the A* algorithm matches Town 01, 02, and 10, showing promising results. On the other hand, in relation to HGA, Town 06 displays a narrow distribution range (approximately 450–550 Wh), suggesting precision in energy usage, and

as for Town 07, the range is observed to be wider with a maximum reaching 1600 Wh, indicating variability but a generally efficient performance.

A real-world application on autonomous vehicles requires low computational times in order to have real-time results. In our system, the more time-consuming functionality is the determination of the optimal route to follow. PPAs, due to continuous search and long iterations depending on the input size, must offer as low execution times as possible. Figure 8 is a comprehensive demonstration of the mean computational time across all maps for both PPAs. A* has consistently low computation times across all maps, ranging from 0.01 to 0.05. By definition, A* is a well-known algorithm for its efficiency and speed in finding a path, although it may not always find the most optimal path if the heuristic is not perfectly suited to the environment. In contrast, the HGA has significantly higher computation times, ranging from over 10 to over 40 s. The inherent structure of the algorithm demands more computational processes, and the fact that the execution time does not drop below 10 s suggests a baseline computational overhead that is significant in comparison to A*.

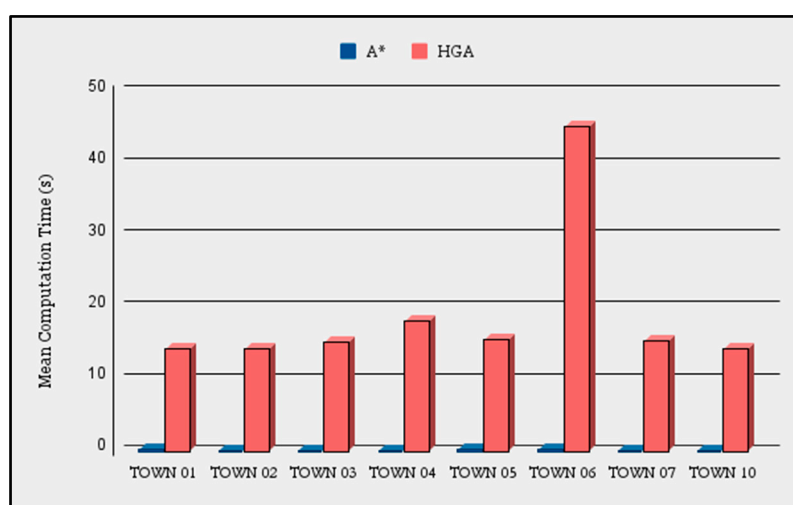


Figure 8. Comprehensive representation of the mean computational times across all applicable maps or both PPAs.

With that being said, HGA's approach, involving complex evaluations and multiple iterations over a population of solutions, aims at optimizing the path quality. This takes considerably longer but yields better results in terms of path efficiency and adaptability to complex environments. The practical implications determine the choice of each algorithm, with the A* having an almost immediate response—thus scenarios such as emergency response navigation—and with the HGA finding an ideal role in applications where the quality of the path is more critical, such as routing for energy efficiency and journey safety or where paths can be precomputed and stored.

The collective understanding of the previous figures proves that the HGA is more suited for complex, scalable environments and longer distances, thus demonstrating superior scalability compared to the A* algorithm. The HGA's use of a dynamic objective function allows for more straightforward adjustments to meet various optimization goals by modifying the function itself. In contrast, A* requires changes to the entire heuristic operation, which can be more complex and less flexible. Nevertheless, it is important to note that both PPAs are dependent on how the graph is structured to meet their operational needs. Therefore, while it cannot be conclusively stated that HGA is the more flexible algorithm overall, it is certainly easier to adjust in response to different requirements.

We present the final figure, Figure 9. This figure consists of two radar plots; the left once again corresponds to the A* performance and the right to the HGA. In relation to the first, increased metrics concentrated on the bottom part of the radar are illustrated. These correspond to Towns 04, 05, and 06. The longest path distances can be found on those maps

with a moderate node count extracted from the algorithm's suggested routes. This means that the complexity in terms of nodes is not excessively high but still significant. Moreover, the central placement of duration values suggests generally low times between nodes and, thus, the lower duration. Adversely, the radar plot for the HGA shows metrics extending in multiple directions, indicating a more uniform distribution of energy consumption, nodes, distance, and durations across all maps. The complexity level of the paths is more consistent in larger maps, and the duration is higher compared to the A*, but in a similar manner, the estimated time to be executed is also somewhat uniform and does not vary widely across distinct map cases.

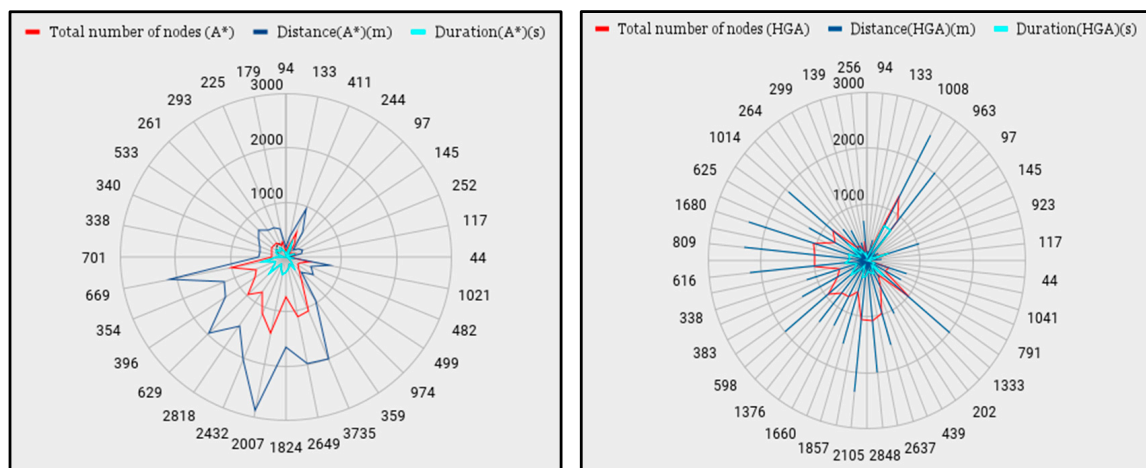


Figure 9. Radar plots that summarize the combination of metrics for the A* (left radar) and the HGA (right radar) across all maps.

5. Conclusions and Next Steps

Our work revolves around the development of a solution framework regarding vehicular networks and electric vehicles that combine an energy estimation model with path planning algorithms to find the optimal route in multiple testing grounds. We aspired from the beginning the conceptualization of our work as a testing tool for different vehicles and different maps, and while the processes around the simulator, such as the selection of depot and destination, can be significantly improved, the capabilities of the framework and its dynamic nature more than cover its flaws. Regarding the path planning algorithms, HGA appears to provide a more balanced approach to pathfinding with a consistent distribution of metrics across the various maps. It is apparent that the algorithm adapts well to distinct environmental conditions and optimizes paths more uniformly for factors, distance, duration, and the population of nodes. The A* shows great efficiency in computation time, particularly in complex towns, but at the cost of potentially longer paths, which might not always be optimal in terms of overall route efficiency. Therefore, the selection of the ideal algorithm really depends on the application and the environmental characteristics of the map. Finally, our implementation of the energy estimation model can closely approach the real-world efficiency of a vehicle, albeit in the best case, as in unfavorable scenarios, it shows a lack of consistency and efficiency. For future work, we are already focusing on improving the energy estimation model, and we aspire to add factors like weather, road conditions, and traffic to resemble real-world conditions. We aspire to leverage the cubic densest packing (CDP), also known as face-centered cubic packing (FCC), a method to create a dense structure where these factors affecting the model are densely and efficiently considered. We are also working on creating a State of Charge (SoC) estimation to address the battery performance of an electric vehicle. Lastly, we are considering adding more path-finding algorithms and improving the overall interface of the framework.

Author Contributions: Conceptualization, D.G., K.K. and G.D.; Methodology, D.G. and I.V.; Software, D.G. and I.V.; Validation, D.G. and T.P.; Writing—original draft, D.G.; Writing—review & editing, K.K., C.B., I.P. and G.D.; Supervision, C.B., I.P. and G.D.; Project administration, K.K. All authors have read and agreed to the published version of the manuscript.

Funding: The project is supported by the Chips Joint Undertaking and its members, including the top-up funding by the national Authorities of Germany, Belgium, Spain, Sweden, Netherlands, Austria, Italy, Greece, Latvia, Finland, Hungary, Romania, and Switzerland, under grant agreement number 101096387. Co-funded by the European Union.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: 3rd Party Data. No data generation of data is applied.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kyaw, P.T.; Le, A.V.; Veerajagadheswar, P.; Elara, M.R.; Thu, T.T.; Nhan, N.H.K.; Van Duc, P.; Vu, M.B. Energy-efficient path planning of reconfigurable robots in complex environments. *IEEE Trans. Robot.* **2022**, *38*, 2481–2494. [\[CrossRef\]](#)
2. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Path planning and trajectory planning algorithms: A general overview. *Motion Oper. Plan. Robot. Syst. Backgr. Pract. Approaches* **2015**, *29*, 3–27.
3. Tang, G.; Tang, C.; Claramunt, C.; Hu, X.; Zhou, P. Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment. *IEEE Access* **2021**, *9*, 59196–59210. [\[CrossRef\]](#)
4. Babu, A.; Kavitha, T.; de Prado, R.P.; Parameshachari, B.D.; Woźniak, M. HOPAV: Hybrid optimization-oriented path planning for non-connected and connected automated vehicles. *IET Control Theory Appl.* **2023**, *17*, 1919–1929. [\[CrossRef\]](#)
5. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An open urban driving simulator. In Proceedings of the 1st Conference on Robot Learning (CoRL 2017), Mountain View, CA, USA, 13–15 November 2017.
6. Liebig, T.; Piatkowski, N.; Bockermann, C.; Morik, K. Dynamic route planning with real-time traffic predictions. *Inf. Syst.* **2017**, *64*, 258–265. [\[CrossRef\]](#)
7. Wang, H.; Qi, X.; Lou, S.; Jing, J.; He, H.; Liu, W. An efficient and robust improved A* algorithm for path planning. *Symmetry* **2021**, *13*, 2213. [\[CrossRef\]](#)
8. Cai, Y.; Xi, Q.; Xing, X.; Gui, H.; Liu, Q. Path planning for UAV tracking target based on improved A-star algorithm. In Proceedings of the 2019 1st International Conference on Industrial Artificial Intelligence (IAI), Shenyang, China, 23–27 July 2019.
9. Kapi, A.Y.; Sunar, M.S.; Zamri, M.N. A review on informed search algorithms for video games pathfinding. *Int. J. Adv. Trends Comput. Sci. Eng.* **2020**, *9*, 2756–2764. [\[CrossRef\]](#)
10. Yin, J.; Fu, W. A hybrid path planning algorithm based on simulated annealing particle swarm for the self-driving car. In Proceedings of the 2018 International Computers, Signals and Systems Conference (ICOMSSC), Dalian, China, 28–30 September 2018.
11. El-Mihoub, T.A.; Hopgood, A.A.; Nolle, L.; Battersby, A. Hybrid Genetic Algorithms: A Review. *Eng. Lett.* **2006**, *13*, 124–137.
12. Osman, I.H.; Laporte, G. Metaheuristics: A bibliography. *Ann. Oper. Res.* **1996**, *63*, 511–623.
13. Maroof, A.; Ayvaz, B.; Naeem, K. Logistics optimization using hybrid genetic algorithm (hga): A solution to the vehicle routing problem with time windows (vrptw). *IEEE Access* **2024**, *12*, 36974–36989. [\[CrossRef\]](#)
14. Zhongrui, Y.; Houyu, Y.; Miaohua, H. Improved Ant Colony Optimization Algorithm for Intelligent Vehicle Path Planning. In Proceedings of the 2017 International Conference on Industrial Informatics—Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), Wuhan, China, 2–3 December 2017; pp. 1–4.
15. Zheng, L.; Yu, W.; Li, G.; Qin, G.; Luo, Y. Particle Swarm Algorithm Path-Planning Method for Mobile Robots Based on Artificial Potential Fields. *Sensors* **2023**, *23*, 6082. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Weerasuriya, A.; Zhang, X.; Wang, J.; Lu, B.; Tse, K.; Liu, C.-H. Performance evaluation of population-based metaheuristic algorithms and decision-making for multi-objective optimization of building design. *Build. Environ.* **2021**, *198*, 107855. [\[CrossRef\]](#)
17. Iyer, V.H.; Mahesh, S.; Malpani, R.; Sapre, M.; Kulkarni, A.J. Adaptive range genetic algorithm: A hybrid optimization approach and its application in the design and economic optimization of shell-and-tube heat exchanger. *Eng. Appl. Artif. Intell.* **2019**, *85*, 444–461. [\[CrossRef\]](#)
18. Levine, D. Application of a hybrid genetic algorithm to airline crew scheduling. *Comput. Oper. Res.* **1996**, *23*, 547–558. [\[CrossRef\]](#)
19. Oğuz, C.; Ercan, M.F. A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *J. Sched.* **2005**, *8*, 323–351. [\[CrossRef\]](#)
20. Chen, Y.; Wu, G.; Sun, R.; Dubey, A.; Laszka, A.; Pugliese, P. A review and outlook of energy consumption estimation models for electric vehicles. *arXiv* **2020**, arXiv:2003.12873.
21. Li, Z.; Su, S.; Jin, X.; Xia, M.; Chen, Q.; Yamashita, K. Stochastic and Distributed Optimal Energy Management of Active Distribution Networks Within Integrated Office Buildings. *CSEE J. Power Energy Syst.* **2024**, *10*, 504–517.

22. Mašek, Z.; Závodník, M. BEV Energy Consumption Estimation for Route Planning. In Proceedings of the 2023 International Conference on Electrical Drives and Power Electronics (EDPE), The High Tatras, Slovakia, 25–27 September 2023.
23. Niranjana, D.; VinayKarthik, B.C. Deep learning based object detection model for autonomous driving research using carla simulator. In Proceedings of the 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 7–9 October 2021.
24. Kaur, P.; Taghavi, S.; Tian, Z.; Shi, W. A survey on simulators for testing self-driving cars. In Proceedings of the 2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD), Detroit, MI, USA, 28–29 April 2021.
25. Shet, R.M.; Iyer, N.C.; Mirje, M.; Bikkannavar, K.V.; Rokhade, S. Path Planning of Autonomous Vehicle for Real World Scenario Using CARLA. In Proceedings of the International Conference on Information and Communication Technology for Competitive Strategies, Jaipur, India, 8–9 December 2023; Springer Nature: Singapore, 2023.
26. Erke, S.; Bin, D.; Yiming, N.; Qi, Z.; Liang, X.; Dawei, Z. An improved A-Star based path planning algorithm for autonomous land vehicles. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420962263. [\[CrossRef\]](#)
27. U.S Environmental Protection Agency (EPA). Data on Cars Used for Testing Fuel Economy. US EPA. Available online: <https://www.epa.gov/compliance-and-fuel-economy-data/data-cars-used-testing-fuel-economy> (accessed on 24 May 2024).
28. Faraz, A.; Ambikapathy, A.; Thangavel, S.; Logavani, K.; Prasad, G.A. Battery electric vehicles (BEVs). In *Electric Vehicles. Green Energy and Technology*; Springer: Singapore, 2021; pp. 137–160. [\[CrossRef\]](#)
29. Archtoolbox. Calculating Slope and Common Slopes in Architecture. Archtoolbox. 2022. Available online: <https://www.archtoolbox.com/calculating-slope/> (accessed on 17 April 2024).
30. Jeddisaravi, K.; Alitappeh, R.J.; Guimarães, F.G. Multi-objective mobile robot path planning based on A* search. In Proceedings of the 2016 6th International Conference on Computer and Knowledge Engineering (ICCCKE), Mashhad, Iran, 20 October 2016; pp. 7–12.
31. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson: London, UK, 2016.
32. Lobo, F.G.; Goldberg, D.E. Decision making in a hybrid genetic algorithm. In Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), Indianapolis, IN, USA, 13–16 April 1997.
33. Elmihoub, T.; Hopgood, A.A.; Nolle, L.; Battersby, A. Performance of hybrid genetic algorithms incorporating local search. In Proceedings of the 18th European Simulation Multiconference, Magdeburg, Germany, 13–16 June 2004.
34. Chuang, L.-Y.; Yang, C.-S.; Yang, H.-S.; Yang, C.-H. Identification of high-order single-nucleotide polymorphism barcodes in breast cancer using a hybrid taguchi-genetic algorithm: Case-control study. *JMIR Med. Inform.* **2020**, *8*, e16886. [\[CrossRef\]](#)
35. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [\[CrossRef\]](#)
36. Dorigo, M.; Maniezzo, V.; Colnari, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [\[CrossRef\]](#) [\[PubMed\]](#)
37. Douglas, D.H.; Peucker, T.K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartogr. Int. J. Geogr. Inf. Geovisualization* **1973**, *10*, 112–122. [\[CrossRef\]](#)
38. Cartography Playground. Douglas-Peucker Algorithm. Cartography Playground. 2021. Available online: <https://cartography-playground.gitlab.io/playgrounds/douglas-peucker-algorithm/> (accessed on 27 March 2024).
39. Electric Vehicle Database. Audi e-Tron 55 Quattro (2019) Price and Specifications. EV Database. March 2019. Available online: <https://ev-database.org/car/1092/Audi-e-tron-55-quattro> (accessed on 24 May 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.