Enhancing Genetic Algorithm with Explainable Artificial Intelligence for Last-Mile Routing

Yonggab Kim, Reem Khir, and Seokcheon Lee

***Abstract*—Traditional evolutionary optimization algorithms often presuppose straightforward constraints and objective functions. However, in many real-world optimization problems, a clear-cut objective may be absent or hard to evaluate. Given these challenges, surrogate models have gained attention as proxies for evaluating objective functions or constraints. In this research, we leverage Machine Learning (ML) based surrogate modeling and Explainable Artificial Intelligence (XAI) to improve the performance of Genetic Algorithm (GA) for route sequencing problem. Our framework utilizes ML to capture the nuanced interplay within environmental data while using XAI to identify their relative importance, to ultimately uncover tacit knowledge embedded in desired solutions. The goal is to use ML and XAI to guide the GA search by identifying and utilizing *significant* genes, i.e., genes that produce high-quality solutions in a more efficient manner. The proposed approach is data-driven, focusing on enhancing core GA components, including (1) enhanced chromosome generation and initialization, (2) informed genetic operators, and (3) refined evaluation of fitness function. We demonstrate our framework using real-world data from the Amazon 2021 Last Mile Challenge as a case study. Our methodology extracts drivers' preferred routing characteristics and applies these insights to generate new routes. Our method is shown to generate high-quality routes when compared to other approaches from the literature, demonstrating improvements in the convergence and effectiveness of GA enhanced by XAI. The code for this publication can be found at https://zenodo.org/records/15227301.**

***Index Terms*—machine learning, explainable AI, genetic algorithm, last-mile routing.**

I. Introduction

GENETIC Algorithms (GA) have gained attention as a popular class of evolutionary algorithms (EAs) due to their effectiveness in solving various constrained optimization problems. GA draws inspiration from the principles of natural selection and genetics process, enabling it to traverse complex solution spaces across various domains; see [1] for a comprehensive review on its theory and applications. GA are known for their adaptability, allowing the exploration of a multitude of potential solutions through mechanisms such as mutation, crossover, and selection. However, GA's performance depends heavily on the definition of the evaluation space—a representation of objective and constraint functions—over which they traverse [2].

In real-world applications, particularly in combinatorial optimization problems, the fitness landscape is often obscured by a lack of explicit objective functions or constraints. To address this issue, Surrogate-Assisted Evolutionary Algorithms (SAEAs) have emerged as an effective alternative, utilizing surrogate models to approximate the fitness function [3], [4] while allowing for less computationally demanding searches [5]. Such surrogate modeling primarily utilizes Machine learning (ML) approaches and combines it with GA to enable effective decision making. ML has proven its potential in effectively identifying patterns and making predictions from historical data, demonstrating a remarkable ability to generalize across various scenarios [6]. However, ML models in standard SAEAs may be characterized with complex architectures, which poses significant interpretability challenges, making it difficult to understand and explain how and why a specific output is obtained.

In genomic research, the interpretability of ML models has proven to be as valuable, if not more so, than the predictions they render, as it can lead to novel insights into genetic processes [7]. This principle of interpretability holds analogous significance in the optimization tasks addressed by GA. Just as the interpretation of genomic data can improve our understanding of foundational principles in genetic mechanisms, interpretation of influential features in optimization tasks can guide improvement in GA's design and processes. This paper proposes an XAI-enhanced GA framework that integrates a surrogate modeling with a GA using XAI to guide the GA's search process in solving a last-mile routing problem, namely, the route sequencing problem.

We consider the route sequencing problem, a specific problem from Amazon Last Mile Challenge in 2021, which involves determining the optimal order of stops where the objective function is not clearly defined [8]. The proposed framework follows two steps: first, it identifies and prioritizes the features that affect the objectives through constructing a surrogate model and employing explainable AI, namely, SHAP analysis. Second, these prioritized features are incorporated into the evolutionary process to enhance convergence and effectiveness. The framework leverages the unique strengths of its components: machine learning and XAI excel in identifying patterns and analyzing the importance of features, while GA efficiently explore the broad combinatorial solution space. Our approach balances exploration and exploitation by combining the surrogate model's data-driven insights with the GA's robust search capabilities.

The contributions of this paper can be summarized as follows: (1) It introduces an XAI-enhanced GA framework for solving sequencing problems with unknown objective functions. The proposed framework leverages insights from ML and XAI to identify influential features, improving GA design and operations, including chromosome encoding, crossover, and mutation strategies. (2) It applies the framework to a last-mile routing problem using real-world data from the 2021 Amazon Routing Challenge, demonstrating its effectiveness in generating routes closely resembling those of experienced drivers. (3) It shows a capability for GAs to adapt and align with human decision-making patterns gained from XAI, paving the way for research and development of interpretable evolutionary algorithms. (4) It highlights, in an ablation study, the value of each XAI-enhanced component in improving the efficiency and effectiveness of GA compared to a standard form.

The remainder of this paper is organized as follows: Section II provides a review of the relevant background. Section III describes our problem setting, and Sections IV and V describe in detail the proposed methodology and adapted framework. Section VI summarizes our experimental results and discussions. Finally, Section VII concludes the paper with a summary of our findings, a discussion of limitations, and future research directions.

II. Background

Evolutionary algorithms, which draw inspiration from natural evolutionary processes, served for many years as a framework for optimization challenges [9], [10]. These methods have seen diverse applications, from retail and supply chains to manufacturing and other domains [11]–[13]. In this section, we review literature related to classical GA used in this study, and the use of machine learning and explainable artificial intelligence in the context of evolutionary algorithms.

A. Genetic Algorithm

Introduced by J.H. Holland in 1992, GA has been established as a robust problem-solving tool that mimics the evolution process, inspired by the biological process of natural selection [14]. The algorithm utilizes chromosome-like structures to represent potential solutions within the search space, subjecting these to selection and genetic operations rooted in biological principles [15]. In a search for optimal solutions, GA begins with a varied population of encoded solutions, evolving them through mechanisms such as crossover and mutation—genetic operators that combine and alter traits, respectively. Fitness evaluations subsequently assess each potential solution. This evaluation is a critical component of the GA, not only to assess but also rank the solutions, guiding the algorithm and influencing which candidates reproduce and are carried forward [16].

Researchers have developed various versions of GA. The categorization of GA is often based on how chromosomes are represented, mainly dividing them into binary, or real-coded GA. Binary-coded GA employ a binary representation to encode chromosomes, which suits discrete problem spaces and has led to the development of specialized genetic operators tailored for this encoding [17]. On the other hand, Real-Coded Genetic Algorithm (RCGA) use real numbers to represent solutions, offering a more direct approach to continuous optimization problems. This format is particularly effective in handling high-dimensional and complex landscapes where precision is crucial. Research in RCGA has focused extensively on enhancing crossover, mutation, and selection operators to address the complexities of real-world problem spaces [14], [18]. These modifications aim to improve convergence rates and solution quality by more accurately modeling the problem's inherent characteristics. Our framework contributes to the body of literature on RCGA by embedding features inside each gene as a vector of real numbers. This allows us to utilize a rich set of realistic information to guide the evolutionary process, improving the algorithm's effectiveness and interpretability.

B. Evolutionary Algorithm and Machine Learning

The evolution of GA has naturally led to the integration of machine learning techniques. The combination of these technologies is opening new pathways in optimization and artificial intelligence, enabling more efficient problem-solving.

1) Data-Driven Evolutionary Algorithm: Surrogate models have been widely used in data-driven EAs to reduce computational costs. These models use evolutionary algorithms as the primary optimization framework and consider surrogate models to accelerate the convergence and/or replace the objective function. Jin et al. [19] provided a comprehensive overview and case study for data-driven evolutionary optimization. They discussed how historical data can be leveraged to drive the evolutionary algorithm process. Chugh et al. [20] proposed an evolutionary algorithm using Kriging models to approximate objective functions, enhancing efficiency and reducing computational costs. The algorithm balanced diversity and convergence, utilizing uncertainty in the Kriging model outputs while considering the distribution of reference vectors and individual positions. Cai et al. [21] introduced a surrogate-guided Genetic Algorithm (SGA) developed for solving high-dimensional, computationally intensive problems using radial basis function. They utilized a surrogate-based trust region local search method for accurate GA search guidance. This was complemented by an SGA updating mechanism with a neighbor region partition strategy to effectively guide the GA's crossover operations. Lin et al. [22] developed an ensemble surrogate-based framework to manage the growing number of decision variables in multiobjective optimization problems. Their approach involved constructing a global surrogate model for exploring the entire search space, accompanied by several surrogate submodels dedicated to exploiting specific sub-areas. Their methodology enhanced the prediction accuracy and reliability in approximating costly objective functions. Other research, including [23] and [24], showed that a surrogate model not only speeds up the convergence but also enhances the overall robustness of the EA in navigating through high-dimensional, complex search spaces.

2) Explainable AI: With the increasing adoption of data-driven decision models in operations research, the ability to comprehend and explain the outcomes of these models is becoming essential. Bock et al. [25] discussed three principles of Explainable AI in Operation Research (XAIOR) that can address operations research challenges and improve decision-making processes. Firstly, incorporating XAI into the methodology as performance analytics enhances the effectiveness and efficiency of decision-making [26], [27]. Secondly, using XAI for attribute analytics allows decision-makers to develop understandable, justifiable, and actionable decisions [28], [29]. Thirdly, applying XAI as responsible analytics ensures legal compliance, ethical responsibility, and frugality in decision-making [30]–[32]. Recent literature on EAs highlighted the importance of incorporating XAI into EAs to enhance interpretability and trustworthiness. Bacardit et al. [33], Wang et al. [34], and Mei et al. [35] provided comprehensive overviews of this emerging topic. These authors illustrated the potential of XAI in aiding EAs to be intrepretable, enabling users to comprehend, trust, and effectively manage these complex algorithms. One of the aspects of XAI in the context of EAs is the visualization of the evolutionary process, which is discussed in depth by Walter et al. [36]. Their work demonstrated how visualization strategies can be used to map the evolutionary journey of solutions, offering stakeholders a window into the dynamic decision-making process of EAs. To simplify EAs' complexity, Fyvie et al. [37] explored the integration of Principal Component Analysis (PCA) with evolutionary algorithms. By reducing the dimensionality of the data, PCA aids in identifying the essential features that drive solution evolution. Another aspect of XAI research is the use of the dominance-based rough set approach to incorporate explainability within EAs. Corrente et al. [38] showcased the application of "if-then" rules to explain and steer the direction of genetic algorithm. This approach enabled the translation of complex evolutionary operations into a set of comprehensible rules, providing a rationale for each evolutionary step.

Two important concepts of XAI are feature importance and feature attribution. Feature importance assesses how much a particular feature contributes to the overall predictive performance of a model [39]. For instance, in a routing problem, factors such as delivery time windows, traffic patterns, and distances between stops can be ranked based on their overall impact. On the other hand, feature attribution focuses on how the presence of a feature impacts individual predictions. For example, SHAP values might reveal that traffic congestion contributes an additional 0.2 minutes to the predicted delivery time for a specific route, while the distance to the next stop reduces it by 0.1 minutes. While feature importance offers an overall ranking of features on a global scale, feature attribution provides specific values that clarify individual predictions. In linear models, feature influence is captured through coefficients. However, nonlinear models require more advanced techniques [40], [41].

A method for integrating both feature importance and feature attribution is the SHAP (SHapley Additive exPlanations) value. SHAP values decompose a model's prediction into the marginal contributions of individual features, offering a unified approach to understanding these predictions. SHAP values possess several key properties that are beneficial for complex nonlinear models. These properties include local accuracy, meaning the sum of the SHAP values for all features equals the model's prediction; missingness, which assigns a SHAP value of zero to features that are not included in the model; and consistency, which guarantees that if a feature's contribution increases, its SHAP value will also not decrease [42].

Although the computation of SHAP values can be expensive, requiring 2^N evaluations for N features, model-specific approximations have been developed to make SHAP values more computationally feasible [42], [43]. Specifically, the SHAP formula is defined as:

$$\phi_j = \sum_{B \subseteq F \setminus \{j\}} \frac{|B|!(N-|B|-1)!}{N!} \left[f_x(B \cup \{j\}) - f_x(B) \right],$$

where N is the total number of features, and F represents the complete set of features. The set B includes all possible subsets of F that do not contain the feature j. The function $f_x(B)$ defines the model's expected output when only the subset B is provided as input [44]. The difference $f_x(B \cup \{j\}) - f_x(B)$ evaluates the impact of adding the feature j to the subset B on the model's prediction. By taking an average of this marginal contribution across all subsets B, the SHAP value ϕ_j captures the influence of feature j on the final prediction.

Our framework falls under data-driven evolutionary algorithms as we utilize surrogate models to guide GA in reducing computational costs and ensuring effective solutions. However, instead of using a black box model, our framework interprets the association built within surrogate models using SHAP values. This interpretation allows us to embed data-driven insights directly into the GA mechanism. This approach provides human users with an understanding of critical factors and ensures GA's solutions are trustworthy and actionable.

III. Route Sequencing Problem

The Route Sequencing Problem (RSP) is defined as follows: Let $\mathcal{M}$ represent the set of stops where each stop $i = 1, \ldots, M$ must be visited exactly once, and a driver can visit only one stop at a time. Stop 1 represents the station, where each route begins and ends. The goal is to find an optimal sequence $S^* = (s_1^*, s_2^*, \ldots, s_M^*)$, with $s_i^* \in \mathbb{Z}^+$ specifying the ordering of each stop i in that sequence.

We consider a setting where the objective function is not well-defined or precisely known in advance. In particular, the stops in the route have a desired ordering influenced by a set of environmental factors that are not completely known in advance; that is, the objective function is not necessarily driven by a single objective known *a priori*. For every stop $i \in \mathcal{M}$, we define $\mathbf{k}^{(i)}$ to denote a vector of N environmental features. Each feature $k_j^{(i)}$ represents the value of the j-th feature influencing stop i, where $j = 1, \ldots, N$. These features include key characteristics of the stop, such as delivery zones, time windows, and package details, which may impact the stop's position in the sequence—an aspect that XAI techniques will allow us to investigate.

The underlying problem of the RSP can be formulated as a Traveling Salesman Problem (TSP) with an unknown objective function. In this work, we adopt the Miller–Tucker–Zemlin (MTZ) formulation of the TSP as described below.

a) Sets, Indices and Parameters:

- $\mathcal{M}$: Set of stops, where $i \in \mathcal{M} = \{1, 2, \ldots, M\}$ denotes a stop in the sequence, with stop 1 designated as the depot.

b) Decision Variables:

- $x_{i,q} \in \{0,1\}$: Binary variable indicating whether stop $q \in \mathcal{M}$ is visited immediately after stop $i \in \mathcal{M}$, $i \neq q$.
- s_i: The position of stop i in the sequence

c) Objective Function:

$$\min \qquad f(x,s) \tag{1}$$

d) Constraints:

$$\sum_{q=1,q\neq i}^{M} x_{i,q} = 1 \quad \forall i \in \mathcal{M}, \tag{2}$$

$$\sum_{i=1,i\neq q}^{M} x_{i,q} = 1 \quad \forall q \in \mathcal{M}, \tag{3}$$

$$s_i - s_q + (M-1)(1-x_{i,q}) \geq 1 \quad \forall \quad 2 \leq i \neq q \leq M, \tag{4}$$

$$2 \leq s_i \leq M \quad \forall \quad 2 \leq i \leq M. \tag{5}$$

The objective function $f(x,s)$ is unknown. The assignment and precedence constraints (2) and (3) ensure that a valid sequence visits each stop exactly once. Subtour elimination constraints (4) and (5) prevent disconnected routes, ensuring a single, continuous route. RSP can be viewed as a special case of the asymmetric Traveling Salesman Problem (ATSP), which is known to be NP-hard. Solving it directly for the scale and composition of instances considered in this case study is impractical, motivating the need for the proposed GA approach.

IV. Proposed Methodology

Our proposed methodology combines machine learning (ML) with a genetic algorithm (GA) to solve the Route Sequencing Problem (RSP), which is combinatorial in nature. This integrated approach leverages ML's predictive power to refine GA's adaptive search capabilities while using explainable AI (XAI) for interpretability, overcoming the limitations encountered when each of the methods is used in isolation. Fig.1 illustrates our proposed framework, which consists of two phases to enhance GA through the integration of machine learning insights.

The initial phase consists of a preprocessing step followed by a Machine Learning surrogate model that learns the association between environmental features and the desired output. This is followed by an explanation model to interpret and unveil the underlying association within the surrogate, black-box model. This allows us to uncover patterns within the data and comprehend the significance of different features in the context of our optimization problem. The second step involves integrating the knowledge derived from ML and XAI directly into the GA, with the aim of accelerating the GA process and refining the fitness function. This strategic combination of machine learning insights with GA enables more efficient and effective optimization search.

A. The Machine Learning Phase

The goal of the machine learning phase is to uncover key features pertinent to a desired solution, which can then be utilized to enhance the computational efficiency of an optimization phase employing evolutionary computations to find high-quality solutions. In particular, the machine learning phase is composed of three core components: (1) a pre-processing step that extracts and prepares data for learning, (2) a ML-based surrogate model that learns a desired solution using historical data; we use surrogate modeling as an *approximation* method that mimics the behavior of an optimization problem with an objective that is not fully known to us, and (3) an explanation model that explains how much each feature contributes to desired outputs; this step involves the use of SHAP, as an XAI feature-attribution method, to explain the predictions of the surrogate ML model and identify the relative importance of features, uncovering tacit knowledge embedded in desired solutions. We discuss each of these components in more detail in what follows.

1) Pre-processing: RSP can be viewed as a function f that, given a set of stops $\mathcal{M}$, returns an optimal (target) solution $\mathbf{s}$ specifying the ordering of each stop in a sequence. A corresponding machine learning model is a function $\hat{f}$ that approximates f, i.e., $f \approx \hat{f}$. Specifically, we consider a machine learning model that is designed to predict part of the sequence, mainly, predicting the next stop from a given stop; This model is found to be more efficient and interpretable than a long-shot sequence inference model [45].

The pre-processing step aims to prepare historical data for learning. It takes as an input a set of desired (optimal) sequences. It then breaks it down into pairs of consecutive stops to better understand the precedence relations that form the skeleton of a desired sequence. This is particularly important in the absence of a clear metric for sequence evaluation. The output of this step is a vector of pairs of stops $\mathbf{p}$ which is used as an input to the machine learning model $\hat{f}$.

Each pair p captures the precedence relationship between two stops that appear in a desired sequence; specifically, $p := (i, i')$, where stop i' appears as an immediate successor of stop i in a given, highly-rated, historical sequence, i.e., $s_{i'} = s_i + 1$ where $i \in \{1, \ldots, M-1\}, i' \in \{1, \ldots, M\}$. Moreover, each pair p is characterized by a set of environmental features $\bar{\mathbf{k}}^{(p)}$, derived from its stops environmental features, $\mathbf{k}^{(i)}$ and $\mathbf{k}^{(i')}$. We denote $\bar{k}_j^{(p)}$ to represent the value of the j-th environmental feature for every pair p, where $j = 1, \ldots, N$.

2) The Learning Model: While tacit knowledge influencing desired solutions is typically hard to collect directly, it can be learned using past knowledge of highly rated historical solutions. To address this challenge, we propose a surrogate model that leverages ML strength in pattern recognition to provide insights into the structure of desired sequences. Such knowledge can be utilized later on to reduce the search space of GA while producing high-quality solutions.

The surrogate model $\hat{f}(\mathbf{p})$ is a binary classification model that predicts the optimal transition between two pairs: one transition that is desired but for an unknown reason and another transition that is selected based on a conventional

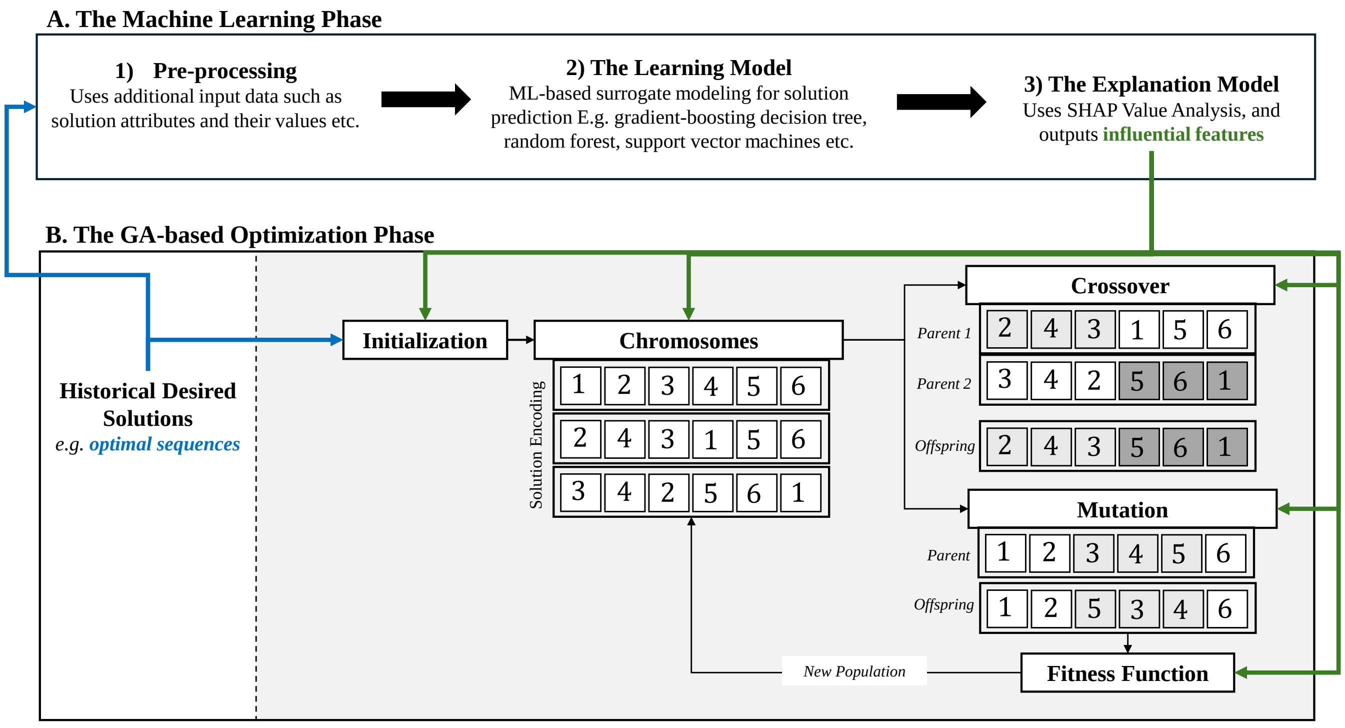
A. The Machine Learning Phase
1) Pre-processing
Uses additional input data such as solution attributes and their values etc.
2) The Learning Model
ML-based surrogate modeling for solution prediction E.g. gradient-boosting decision tree, random forest, support vector machines etc.
3) The Explanation Model
Uses SHAP Value Analysis, and outputs influential features
B. The GA-based Optimization Phase
Historical Desired Solutions
e.g. optimal sequences
Initialization
Chromosomes
Solution Encoding
Crossover
Parent 1
Parent 2
Offspring
Mutation
Parent
Offspring
New Population
Fitness Function

Fig. 1: Illustration of the proposed XAI-Enhanced Genetic Algorithm framework

optimization approach, e.g., minimum distance in the case of last-mile routing. We use $y^{(p)} \in \{1, 0\}$ to denote the output. Note that we randomly label transitions to avoid bias in the evaluation. The model $\hat{f}$ is trained, using a logistic loss function, on a dataset of instances $\{(\mathbf{p}_i, \mathbf{k}_i, \mathbf{y}_i^*)\}_{\{i \in \mathcal{N}\}}$, where $\mathcal{N}$ is the set of instances, $\mathbf{p}_i$ and $\mathbf{k}_i$ are the inputs of instance i representing each pair and their corresponding environmental features, respectively, and $\mathbf{y}_i^*$ is the desired sequence output of instance i.

3) The Explanation Model: While it is often possible to have models that are explainable on their own, complex models, like deep neural nets and ensemble methods, are not easy to understand on their own, requiring simpler *explanation* models for interpretation. Machine Learning interpretation can be classified into two types: local and global. Local interpretation involves understanding a model's prediction for a single instance, while global interpretation involves understanding how a model makes predictions across all inputs. In this work, we utilize *local* methods using Shapley value estimation.

Classic SHAP values quantify the contribution of each feature j to the prediction made by a black-box learning model $\hat{f}$. This is done by creating perturbations of a specific instance in the dataset and observing how these perturbations influence the model's output probability for the positive class. This, in turn, requires retraining the model $\hat{f}$ on all feature subsets $B \subseteq F$, where F is the set of all features.

Each SHAP value, denoted as $\phi_j^{(p)}$, offers an explanation of how the probability of the model predicting a positive class for $y^{(p)}$ is affected by including or excluding feature j; that is, by training a model $\hat{f}_{B\cup\{j\}}$ with feature j present, and another model $\hat{f}_S$ with feature j withheld. A positive $\phi_j^{(p)}$ indicates that the feature increases the probability of the positive class, while a negative value suggests a decrease. The magnitude of $\phi_j^{(p)}$ reflects the strength of the feature's impact in either direction. When these SHAP values are aggregated over many instances, SHAP values can offer global insights into the general patterns of feature importance across the dataset [42]. The aggregated SHAP value [44] for each attribute can be defined using Equation 6.

$$\Phi_j = \frac{1}{M-1} \sum_{p \in \mathcal{P}} |\phi_j^{(p)}| \quad \text{for all } j = 1, \dots, N \tag{6}$$

We apply SHAP value analysis to the surrogate learning model presented earlier, to quantify each feature's influence on the desired solution and provide a rank of its importance. With this enriched dataset, we are positioned to proceed to the next stage: the strategic application of these insights within the GA's framework to refine its search for optimal solutions.

B. The GA-based Optimization Phase

We leverage insights obtained from the ML phase to expedite the GA's convergence and enhance its efficiency and effectiveness. In particular, the goal is to use ML insights to guide the GA search by focusing on *significant* genes, i.e., genes that produce high-quality solutions in a more efficient manner. Algorithm 1 presents the corresponding pseudocode of the proposed XAI-enhanced GA. The proposed approach is data-driven, focusing on core GA components. More specifically, it includes (1) enhanced chromosome generation and initialization, (2) informed genetic operators, and (3) refined evaluation of fitness function, as discussed in more detail in what follows.

Algorithm 1 XAI Enhanced Genetic Algorithm

```
    Input: Problem Data, SHAP Insights
    Output: Optimized Solution
 1: Cluster genes based on SHAP insights
 2: Generate initial population informed by SHAP
 3: for each generation until terminal condition do
 4:     Evaluate fitness using SHAP-informed estimation
 5:     Select elites
 6:     $NewPopulation \leftarrow elites$
 7:     while len($NewPopulation$) < len($Population$) do
 8:         Perform SHAP-informed crossover
 9:         Apply SHAP-informed mutation
10:     end while
11:     $Population \leftarrow NewPopulation$;
12: end for
13: Calculate the fitness of the population
14: return best found solution
```

1) Enhanced Chromosome Generation and Initialization: Key features and their contributions, as identified by the ML phase, can be used to improve the GA encoding for enhanced performance. In this regard, we focus on two aspects. (1) *Gene clustering*: We implement a gene clustering mechanism that organizes genes into coherent groups based on their impact and interactions, as identified through SHAP values. Such clustering reduces the complexity of the search space, resulting in a more effective chromosome encoding while allowing for a more focused exploration that is likely to yield better results in fewer generations. (2) *Initialization*: Traditional GA often relies on simple heuristics for initialization such as random or Nearest Neighbor algorithms. Our goal is to refine this step by integrating insights from the ML phase that identifies desired characteristics of desired solutions to generate initial chromosomes that better align with the desired outcome.

2) Informed Genetic Operators: We leverage the ML phase output to refine two key GA operators: crossover and mutation. Such refinement helps in reducing the search space, and therefore, enhancing the GA's efficiency by eliminating less promising paths and focusing on areas with higher potential.

The *crossover* operator blends segments from parent solutions to generate offspring. This process is instrumental for combining valuable traits from different solutions to explore new parts of the solution space. Leveraging SHAP value analysis, we enhance the crossover operator by *prioritizing* the exchange of genes identified as significantly impacting desired outcomes. For instance, in a GA tasked with optimizing a complex multi-dimensional function, SHAP analysis pinpoints certain dimensions (genes) as particularly influential on the function's value. Using this knowledge, we can fine tune the crossover strategy to ensure these important genes are exchanged more frequently between parents, i.e., giving higher crossover probability to genes recognized as important. This deliberate focus ensures that offspring are more likely to inherit combinations of traits that lead to improved fitness.

The *mutation* operator introduces genetic diversity, i.e. variability, into the search process by randomly altering genes within a solution. Such diversity helps the algorithm avoid stalling and continue to explore a wide spectrum of potential solutions while preventing premature convergence within GA. SHAP value analysis enables us to prioritize diversity in traits identified to have a significant impact on desired outcomes. This targeted approach to diversity ensures that our efforts are concentrated on the most important variations, enhancing the algorithm's ability to navigate and optimize within the solution space effectively.

3) Surrogate-Assisted Fitness Estimation: A key enhancement of GA through our framework is refining the evaluation function. We leverage the output of the ML phase to define a fitness function that captures the underlying desired fitness landscape, not to approximate its evaluation for computational efficiency, but rather to identify relevant objectives driving a desired solution that are not necessarily known in advance. Through SHAP value analysis, we identify and prioritize features that significantly impact the solution's quality. This analysis enables constructing a fitness function that better represents the true nature of the optimization problem. By integrating these key features into the fitness function, we ensure that the GA is guided by an improved understanding of what constitutes a desired solution in the given context.

V. CASE STUDY

RSP considers planning a new set of high-quality delivery stop sequences based on routes that have been historically observed [46]. Distinct from traditional route optimization problems, which rely on well-defined objective functions, such as minimizing travel time, distance, or tardiness, the RSP in this challenge does not have a well-defined objective. Rather, it is governed by a form of tacit knowledge of drivers (domain experts) embedded in highly-rated historical routes [47]. Amazon has acknowledged that drivers often deviate from the routes suggested by its system. Drivers possess knowledge about challenging roads, stops that can be grouped together, peak traffic times, and numerous other aspects that are not explicitly captured by traditional optimization algorithms used for route generation. Such deviation is not desired in practice as it has impact on the accuracy of delivery time windows communicated to customers which is important for customer experience and satisfaction. The goal is to design a solution approach that capture such implicit knowledge, aiming to produce high-quality routes that are as close as possible to drivers' best practices.

A. Available Data

We use an open-source database published as part of the Amazon Last Mile Routing Research Challenge in 2021 [8]. The dataset includes information about Amazon's last-mile delivery operations, encompassing more than 6,100 historical routes across five major US cities: Boston, Austin, Chicago, Los Angeles, and Seattle.

Each route starts at one of 17 distribution depots located in these cities and has a *recommended* sequence of delivery stops a driver needs to visit over the course of a day. The number of stops on a route ranges from 32 to 237 stops, with an average of 148 stops per route. Every stop represents the actual parking spot of the driver, and it has package-related information, including the count, dimensions, and the anticipated service duration associated with it. Additionally, each stop has other information including its zone identification, delivery time windows, and geographic coordinates. The dataset also includes calculated travel times between each stop; the time it takes to get from one stop to another is different in both directions due to varying traffic conditions, making the travel times between stops asymmetric. For each of the 6,112 routes, the dataset includes the order of stops that the delivery drivers *actually* took along with a rating that reflect the quality of the route; these driver routes are rated as either high (2,718 routes), medium (3,292 routes), or low (102 routes) quality routes by Amazon, though the criteria for these classifications have not been disclosed.

It is important to note that these routes were provided to participants for analysis during the competition. Additionally, Amazon published an evaluation set of 3,052 high-quality routes to assess model performance and solution quality. This evaluation set was not revealed during the competition but was made available after it ended for testing purposes. This evaluation set allows for a comparison between the proposed routes and the actual routes taken by drivers, providing a benchmark for scoring the proposed solutions.

To ensure consistency and fairness in our experimental setup, we use the 2,718 high-quality routes for the machine learning phase in our framework as initially published. This dataset is used for training, validation, and testing data to assess the machine learning model's performance. We perform a SHAP analysis on the model created from this dataset to uncover data patterns and determine feature importance. Based on these findings, we fine-tune the optimization phase of our framework. The genetic algorithm of the optimization phase is tested on the evaluation set comprising 3,052 routes. These results are compared with published benchmarks, as discussed later in Section VI-B.

B. Evaluation Metric

Amazon proposed a scoring metric to evaluate the effectiveness of the proposed approaches. We use this metric to evaluate our proposed routes from the optimizer phase.

The scoring metric, referred to as *normalized edit distance similarity* metric, combines two factors to measure the difference between two routes: the positions of the stops in routes, and the travel time between stops within routes. Edit distance, also known as Levenshtein distance, quantifies the number of edit operations required to transform one sequence into another, calculating the minimum cost. This cost is determined by summing up the least expensive combination of weighted operations, which include insertions, deletions, and substitutions. More formally, the proposed metric is defined using equation 7 as follows.

$$\psi(\mathcal{S}_1,\mathcal{S}_2)=\frac{SD(\mathcal{S}_1,\mathcal{S}_2)-ERP_{\text{norm}}(\mathcal{S}_1,\mathcal{S}_2)}{ERP_{\epsilon}(\mathcal{S}_1,\mathcal{S}_2)} \tag{7}$$

Here, $\psi(\mathcal{S}_1,\mathcal{S}_2)$ represents the similarity score of two routes, SD represents the sequence deviation, ERP_{norm} is the normalized edit distance with a real penalty accounting for normalized travel times, and ERP_{ϵ} signifies the number of edit operations performed. When the two routes $\mathcal{S}_1$ and $\mathcal{S}_2$ are identical, $\psi(\mathcal{S}_1,\mathcal{S}_2)$ defaults to zero, reflecting the absence of deviation. Details of the scoring algorithm and related implementation can be found at https://github.com/MIT-CAVE/rc-cli/tree/main/scoring.

C. Adapted Methodology

1) The Machine Learning Phase: The ML phase aims at learning features that contribute to a desired solution. In the context of RSP, this corresponds to learning features that influence a driver's choice of delivery route beyond common metrics, like shortest distance or fastest route.

Pre-processing: First, we pre-process 2,718 high quality routes, breaking each down to capture the precedence relationship between stops within a route and construct $\mathbf{p}$ as vector of pairs of stops for each such instance i. We also extract all the features and save their associated values in $k^{(i)}$.

Learning Model: We build a binary classification model as a surrogate model that predicts whether a transition from one stop to another at the pair level is based on an apparent proximity logic or is due to another unknown reason. In doing so, a *gradient-boosting decision tree* model is employed; this model demonstrated superior performance in comparison to the other classification models we tested, namely random forest and support vector machines. The dataset of $2,718$ instances (high-quality routes) was split into 80% for training and 20% for testing the machine learning phase. We conducted 5-fold cross-validation on the training set to find the best hyperparameters, as detailed in Appendix C. The model was trained using the logistic loss function, achieving an accuracy of almost 80%.

It is important to note that identifying a comprehensive list of potential environmental factors typically requires domain knowledge about the problem at hand. TableI summarizes the environmental factors considered in our study. Note that these factors are considered for each of the stops; the stop in the *actual* sequence, and the stop that is based on a typical known objective; minimum transit time in our case.

Explanation Model: We use the outputs of the learning model as inputs to SHAP for model explanation. Fig.2 summarizes some of the feature's contribution to the model's predictions. A complete SHAP summary plot with all the features can be found in the Appendix A. This analysis involves retraining the surrogate model by permuting the data for one feature at a time within the model while other features remain constant to observe how predictions change. It illustrates the effect of individual features on the predicted outcomes across multiple instances, with each instance represented by a dot.

In this plot, every dot corresponds to the SHAP value of a feature for a particular instance. The y-axis shows features in decreasing order of importance, based on their Φ_j values, while the x-axis displays the extent of each feature's influence on the model's predictions. The magnitude of a SHAP value

TABLE I: Description of input features

Feature	Description
SameZoneAsPrevious[i]	A binary indicator where 1 denotes that stop i has the same zone ID as its previous stop (within-zone stop transition), and 0 indicates a different zone ID (cross-zone stop transition).
TransitTime[i]	The estimated time required to travel to the stop i from its previous stop.
ZoneSimilarity[i]	A measure of similarity of the stop i's zone ID to its previous stop's zone ID.
ServiceTime[i]	The expected duration to complete the service at a stop i.
AvgServiceTimePerPackage[i]	The average expected time to service each package at a stop i.
PackageCount[i]	The total number of packages to be delivered at a stop i.
TotalVolumeCm3[i]	The cumulative volume of all packages to be delivered at a stop i.
VolumeCapacityRatio[i]	The ratio of stop i's package volume to the delivery vehicle's total capacity.
UrgencyStart[i]	The difference in time between the executor's departure and the start of the time window of stop i.
UrgencyEnd[i]	The difference in time between the executor's departure and the end of the time window of stop i.

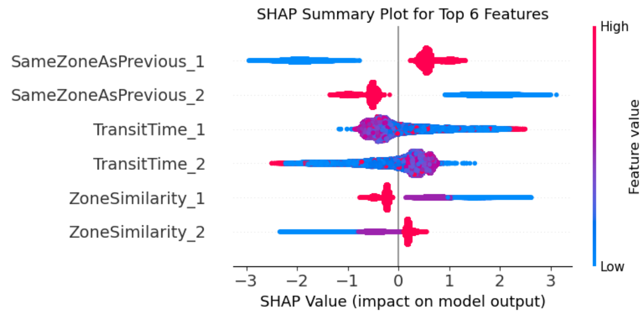

Fig. 2: SHAP summary plot including the most significant features

demonstrates the effect of the feature on the binary classification result. For instance, a SHAP value positioned to the right of the centerline signifies the feature's positive influence on the probability of the outcome being in a positive class; a positive class in our context refers to a transition to the optimal stop, the preferred stop that appears in a desired sequence. In contrast, a value to the left suggests a contribution towards the negative class. The dot color indicates the feature's value in a specific instance, with red indicating higher values and blue indicating lower values.

Moreover, the spread of the points for each feature reflects the variability of its impact across the dataset: a wider spread means that a feature has a varying influence on different predictions while a narrower spread indicates a more consistent influence. Fig.3 summarizes the relative importance of each feature, using SHAP values averaged over all instances. Note that features including SameZoneAsPrevious, ZoneSimilarity, and TransitTime collectively account for 92% of the total SHAP value, which we discuss in more detail in what follows.

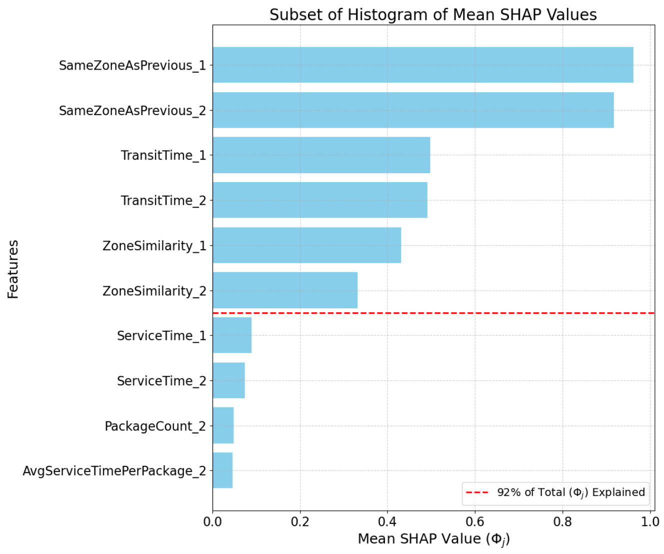

Fig. 3: Mean SHAP Value Histogram

- *SameZoneAsPrevious* This feature indicates whether the transition to a next stop has the same zone ID as the previous stop; recall that we have two possible transitions: one transition representing the actual (desired) sequence, and the second transition is based on minimum travel time. From Fig.2, observe that this feature forms two distinct color clusters, each positioned on opposite sides of the centerline. This implies the following: If the next stop has the same zone ID as the previous stop, the likelihood that it is the desired stop is substantially increased. In contrast, if the next stop has a different zone ID than the previous stop, the likelihood that it is the desired stop is significantly decreased.
- *ZoneSimilarity* This feature encodes information about the geographic area into which a delivery stop is located; such zoning is common in urban areas [8]. For instance, in the Amazon dataset, an example zone ID is "M-10.3D"; the value before the dash 'M' denotes a specific high-level planning areas, and the value after the dash '10.3D' denotes a particular sub-region planning areas. Fig.4 shows an example of a route, color coded by their zone ID; Interactive visualization is available on https://yg212.github.io/Amazon/.
  Leveraging Amazon's encoding structure for the zone ID, we segment this feature into four distinct components: the value preceding the dash, the value following the dash, the numerical value after the dot, and the alphabetic character following the dot; for example, the zone ID "M-10.3D" is segmented into M, 10, 3, and D. We then calculate what we call a *zone similarity* score using a weighted sum based on the number of matching segments. Notice in Fig.2, this aggregated feature forms connected clusters of dots displaying a gradient from red to blue. This gradient suggests that the likelihood of selecting that stop increases as zone similarity increases. Conversely, a decrease in similarity is associated with a reduced likelihood

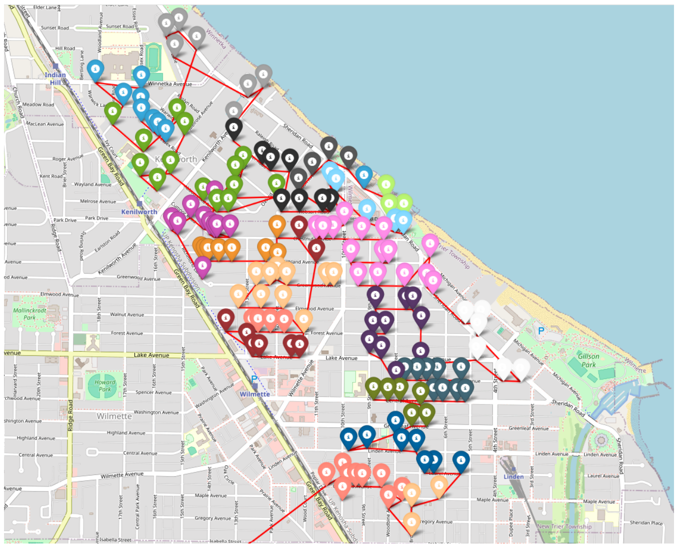
Fig. 4: An example of a high-quality route in the Amazon dataset. Note that consecutive stops tend to belong to similar zones, with each zone represented by a different color.

of selection. *This feature serves as a gradient metric, refining the binary nature of the 'SameZoneAsPrevious' feature*. More specifically, while 'SameZoneAsPrevious' indicates whether consecutive stops share the same zone ID, 'Zone Similarity' measures the degree of similarity between zones, effectively capturing the nuances of transitions between stops with different zone IDs.

- *TransitTime* While minimizing transit time is a common objective in the routing literature, it is not necessarily the key influential factor in forming any good route, especially when operating in urban, dense environments similar to those experienced in last-mile delivery. Research shows that driving time accounts for at most 15% of the total route duration when delivering 200-300 parcels, with an average stop-to-stop distance of less than 100 meters [48]. However, after performing our SHAP analysis on the surrogate model, it was found that, transit time is in fact an important feature of desired routes in the given case study, ranked highest after zone similarity.

Notice that, contrary to our intuition, the SHAP analysis indicated that features related to the *time windows* and *package details* had minimal impact on predicting a desired route. As an additional test, we conducted a LIME (Local Interpretable Model-agnostic Explanations) analysis to compare with SHAP analysis. Both methods identified zone ID-related features and transit time as critical, while time window and package information had minimal impact. The LIME plot with top features can be found in the Appendix B.

2) The GA-based Optimization Phase: We utilize the insights obtained from the ML phase to expedite the GA's convergence and enhance its efficiency in finding high-quality solutions. In particular, we utilize key features that were found important to forming a desired sequence. To do so, we focus on leveraging information related to *zone similarity* and *transit time* to enhance the GA design, mainly through enhancing the chromosome encoding, algorithm initialization, operators design, and the fitness function for evaluation.

Fig.5 illustrates a classical architecture that encodes a sequence solution into a chromosome for GA. Each chromosome represents a route, and each gene represents a stop. These genes are instantiated as classes containing comprehensive stop-related data; such design facilitates utilizing stop-level information during the evolutionary process.

(a) *Enhanced Chromosome Generation:* Fig.5 illustrates an example sequence encoding when using classical GA approaches. We enhance such encoding to capture the significance of zone IDs in forming a desired delivery route. To do so, we adjust the chromosome architecture to represent the grouping of stops by their zone IDs, as depicted in Fig.6. This helps generate solutions that prioritize visiting stops within the same zone before moving on to stops outside of the zone. It is important to note that building chromosomes following this clustering technique aids in reducing the search space; it facilitates the use of customized crossover and mutation techniques, as detailed further in later sections.

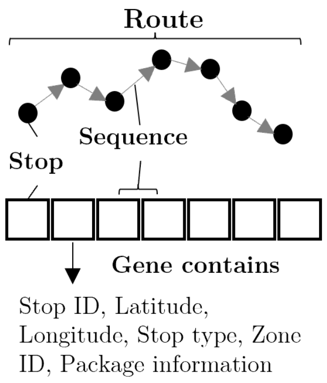
Route
Sequence
Stop
Gene contains
Stop ID, Latitude, Longitude, Stop type, Zone ID, Package information

Fig. 5: illustration of a typical chromosome that encodes a sequence solution with genes indicating delivery stops.

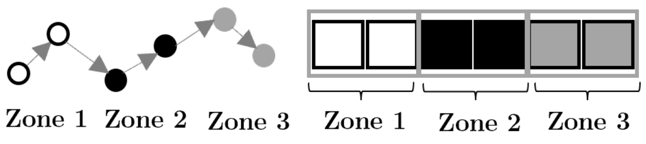

Fig. 6: Enhanced chromosome encoding using zone-based clustering.

(b) *Enhanced Initialization:* We utilize both *zone similarity* and *transit time* measures to generate initial solutions that are diverse to improve the quality of generated solutions; recall, these features were found to be highly influential in forming a desired route. The initialization algorithm utilizes two hierarchical greedy approaches to generate the initial solutions. At the *zone* level, two random weights for transit time and zone similarity are generated. Each unvisited zone is scored based on the weighted sum of the zone similarity score and transit time based on the current zone. The zone with the highest score is selected as the next zone. This process is repeated to create the zone sequence. At the *stop* level, the nearest neighbor algorithm is applied within each zone to optimize the sequence of stops.

(c) *Enhanced Crossover:* Our GA employs a crossover mechanism across two distinct hierarchical levels, as shown in Fig.7: the zone level and the stop level.
At the *zone* level, the Partially Mapped Crossover (PMX) method is employed. The purpose of utilizing PMX is to ensure that the traits of the parent's zone sequence are preserved during the crossover process. PMX crossover maintains the relative order of stops while allowing for the exchange of zone segments between parent chromosomes; this method maps a segment from one parent to the other while using a mapping of indices to resolve conflicts, thus, retaining the original ordering and positioning of stops to a high degree.
At the *stop* level, we use a *Intrazone* crossover strategy. This approach targets reorganizing stops within the same zone to improve route efficiency. Swapping stops that share identical zone IDs between parent chromosomes facilitates the exploration of intra-zone stop arrangements without altering the overall zone sequence structure.

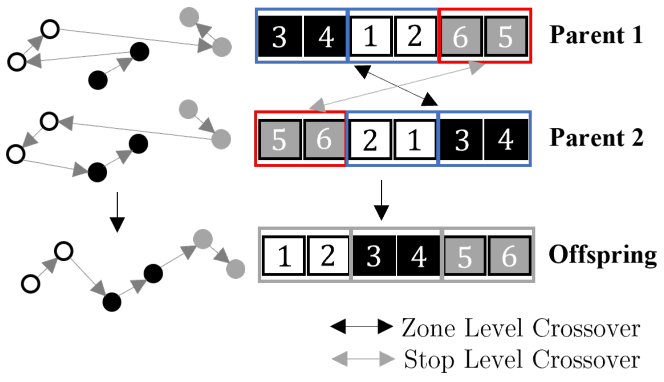

Fig. 7: Enhanded Genetic Algorithm crossover

(d) *Enhanced Mutation:* Our GA incorporates a two-phased mutation strategy, as shown in Fig.8. Similar to crossover, mutation operates at both the zone and stop levels, to introduce variability and enhance the solution space exploration capabilities of the algorithm.
At the *zone* level, two mutation strategies are employed: Reverse and Scramble. The *Reverse* mutation selects a segment within the chromosome and reverses the sequence of zones within this segment. By potentially altering the direction of the route, this mutation strategy can unveil new pathways that may circumvent local optima, enhancing the exploration capabilities of the genetic algorithm. The *Scramble* mutation, on the other hand, randomly shuffles the zones within a selected segment. This introduces high variability into the chromosome, offering a powerful tool for escaping local optima.
Similarly, at the *stop* level, both *reverse* and *scramble*

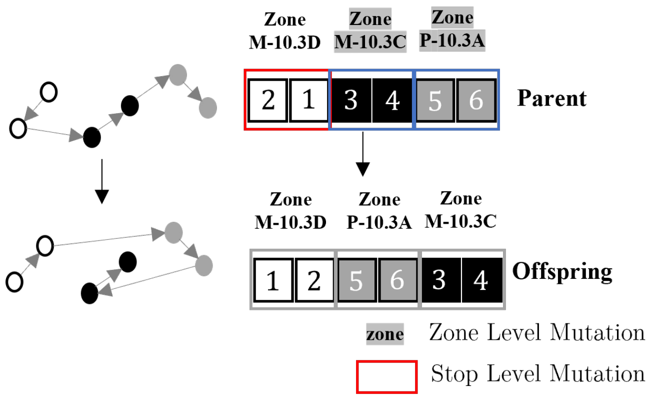

Fig. 8: Enhanded Genetic Algorithm mutation

mutation strategies are applied to individual stops within a zone. The *Reverse* mutation targets a contiguous sequence of stops within a zone and reverses their order. This mutation can reveal more efficient routing options within a zone by exploring alternative directions that may not have been considered during initial route planning. The *Scramble* mutation, on the other hand, randomly permutes the stops within a chosen zone segment. This technique is particularly effective for introducing new combinations of stop sequences that may offer solution improvements.

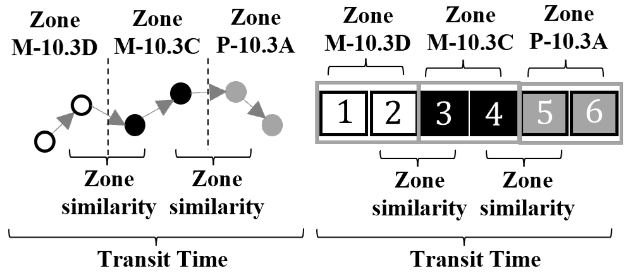

Fig. 9: Illustration of zone similarity and transit time as sequence features used in the evaluation function

3) Evaluation Function: Based on SHAP results, we propose an evaluation function that captures both zone similarity and transit time as important objectives. Specifically, we use a multi-objective fitness function that minimizes the total transit time and maximizes the total zone similarity. Transit time and zone similarity are measured for each stop in relation to its preceding stop, beginning from stop 2 up to stop M. If the zone ID of the previous and current stops are identical, the zone similarity score is set to 0. The evaluation function is illustrated in Fig.9, and it is defined in the following equation:

$$E(\mathcal{S}) = w_1 \cdot \frac{\sum_{i=2}^{M} \text{ZoneSimilarity}[i] - \text{ZoneSimilarity}_{\text{Min}}}{\text{ZoneSimilarity}_{\text{Max}} - \text{ZoneSimilarity}_{\text{Min}}} - w_2 \cdot \frac{\sum_{i=2}^{M} \text{TransitTime}[i] - \text{TransitTime}_{\text{Min}}}{\text{TransitTime}_{\text{Max}} - \text{TransitTime}_{\text{Min}}}$$

where w_1 and w_2 are the weights of each sub-objective. $\text{ZoneSimilarity}_{\text{Min}}$ and $\text{ZoneSimilarity}_{\text{Max}}$ represent the minimum and maximum zone similarity scores from high-quality historical routes, while $\text{TransitTime}_{\text{Min}}$ and $\text{TransitTime}_{\text{Max}}$ correspond to the minimum and maximum transit times. These values are used to normalize the objectives.

The weight of each sub-objective w_1 and w_2 could be determined based on the SHAP analysis. Each sub-objective consists of multiple components, such as ZoneSimilarity_1 and ZoneSimilarity_2. The SHAP contributions of these components are aggregated within their respective categories to compute category totals. These totals are then normalized by dividing each category total by the sum of all category totals, resulting in relative importance scores for each sub-objective. This process results in normalized weights of $w_1 = 0.4355$ for zone similarity and $w_2 = 0.5645$ for transit time. TableII summarizes this process.

TABLE II: SHAP-derived weights for fitness function

Feature	SHAP Contribution	Category Total	Weight
ZoneSimilarity_1	0.43128	0.76324	0.4355
ZoneSimilarity_2	0.33196		
TransitTime_1	0.49795	0.98918	0.5645
TransitTime_2	0.49123		

VI. Computational Results

In this section, we summarize results of our computational experiments. The objectives of these experiments is to (1) demonstrate the effectiveness of the proposed framework when compared to other approaches proposed in the literature, and (2) conduct an ablation study to show the benefits of each of the enhanced components within the GA framework in terms of both efficiency and effectiveness.

A. Experimental Setup

All algorithms were coded in Python, and all experiments were carried out on a server with an i7-12700H 2.30 GHz processor and 32GB RAM. The proposed routes from the optimization phase were assessed using the set of 3,052 evaluation routes. Parameters' settings related to GA implementation are presented in Appendix D. We set a time limit of 12 hours for model building and 4 hours for model application, as specified by the Amazon challenge. Maintaining these time limits was important to ensure consistency with the other approaches we benchmarked against.

B. Performance Evaluation

We conducted a grid search to evaluate how fitness function weights influence algorithm performance. Appendix E summarizes the results, comparing grid search outcomes with weights derived from SHAP analysis. Extreme weight settings, such as $w_1 = 0, w_2 = 1$ (distance-only, score: 0.0553) or $w_1 = 1, w_2 = 0$ (similarity-only, score: 0.0621), led to suboptimal or unrealistic route sequences. Balanced configurations, particularly $w_1 = 0.4, w_2 = 0.6$ and $w_1 = 0.5, w_2 = 0.5$, achieved the best scores of 0.0410. Weights derived from SHAP analysis ($w_1 = 0.4355, w_2 = 0.5645$) achieved a similar score of 0.0411. While slightly less optimal than the best grid search result, the SHAP-based approach can provide key advantages: it can narrow the search space and offer insights into the contribution of each sub-objective.

We further evaluated our algorithm's performance against the leaderboard published by Amazon post competition [49]. The evaluations of the competition were run on an AWS EC2 m5.4xlarge server equipped with an eight-core processor and 16 virtual cores. TableIII provides a summary of the top performers in the competition, including the performance of the proposed XAI-enhanced GA, the neural network-based GA, and a standard version of GA, which we implemented and tested for comparison. The performance metric used in the leaderboard is based on the *normalized edit distance similarity* metric, using the formula described in equation7; it reflects how closely the suggested routes from participants matches the desired highly-rated sequences.

As can be seen in TableIII, our proposed XAI-enhanced algorithm achieved a score of 0.0411, placing it fourth on the leaderboard. It is important to note that other top performing approaches utilized primarily established TSP solution strategies to solve the problem. For example, the top three teams used methods such as Lin-Kernighan-Helsgaun (LKH) [50] implemented in C, Hierarchical Integer Linear Programming [51] in Julia, and Edge Assembly Crossover GA [52] in C++.

Our additional benchmark of a Surrogate-Assisted GA utilizes a neural network to predict the score of a route based on aggregated features, such as total transit time. The network consists of four hidden layers that use ReLU activation, and it was trained on historical data with a Mean Squared Error loss function. While this approach has generality for predicting routes with varying numbers of stops and packages, it may overlook specific details about individual stops, resulting in a score of 0.0790. Similarly, our benchmarking of a standard GA approach achieved a score of 0.1066, which is significantly lower than the score of the XAI-enhanced version. This shows the significance of leveraging data and XAI to enhance standard GA mechanisms and effectively solve hard combinatorial optimization problems like RSP.

C. Ablation Study

We analyzed different variants of our genetic algorithm, examining the incremental benefits of integrating XAI-enhanced components. In particular, we compare the performance of the following variants:

- XAI-Enhanced GA: Genetic Algorithm utilizing XAI-enhanced components for chromosome generation, initialization, and evaluation.
- XAI-Enhanced GA (~~XAI-Initial~~): Genetic Algorithm utilizing XAI-enhanced components for chromosome generation and evaluation only, excluding enhancement in initialization.
- XAI-Enhanced GA (~~XAI-Eval~~): Genetic Algorithm utilizing XAI-enhanced components for chromosome generation and initialization only, excluding enhancement for evaluation.
- XAI-Enhanced GA (~~XAI-Initial~~, ~~XAI-Eval~~): Genetic Algorithm utilizing XAI-enhanced components for chromosome generation only.

TABLE III: Performance evaluation of RSP given Amazon leaderboard [49].

Team Name	Rank	Score	Methodology	Run Times (seconds)	
				Model Build Phase	Model Apply Phase
Just Passing Through	1	0.0248	Constrained Local Search (Lin-Kernighan-Helsgaun)	109	12,095
Permission Denied	2	0.0353	Hierarchical Integer Linear Programming	27	1,825
Sky is the Limit	3	0.0391	Edge Assembly Crossover GA	6	12,080
Proposed XAI-Enhanced GA	**4**	**0.0411**	Proposed XAI-Enhanced Genetic Algorithm	**6**	**12,268**
MEGI	5	0.0436	Zone Prediction with Heuristics	566	2,128
UPB	6	0.0484	-	5	10,434
T-Lab	7	0.0485	-	10	1,174
Turkish Churrasco	8	0.0498	-	2,109	4,873
***Benchmark:* Surrogate-Assisted GA**	21	0.0790	Neural Network-Based Fitness Function GA	30	10,850
Fortaleza	22	0.0911	-	43,203	616
AIDA	23	0.0929	-	4	14,100
Alpha Centauri	24	0.1058	-	10	485
***Benckmark:* Standard GA**	**25**	**0.1066**	Standard Genetic Algorithm	**6**	**12,010**

Note: Methods noted as '-' indicate that the methodology information was not available.

- Standard GA: Genetic Algorithm using standard components without any XAI enhancements.

TABLE IV: Performance comparison of ablated models

Algorithm Variant	Run Times (s)	Competition Score
XAI-Enhanced GA	**12,268**	**0.0411**
XAI-Enhanced GA (~~XAI-Initial~~)	12,589	0.0513
XAI-Enhanced GA (~~XAI-Eval~~)	11,436	0.0553
XAI-Enhanced GA (~~XAI-Initial~~, ~~XAI-Eval~~)	13,514	0.0762
Standard GA	12,010	0.1066

As can be seen in TableIV, just by using XAI to improve chromosome encoding, the performance of GA improved from 0.1066 in its standard form to 0.0762. Refining the evaluation function with XAI insights on top of the enhanced encoding approach yielded the most substantial gains, achieving a score of 0.0513. This further highlights the importance of informed fitness assessments in GA. In addition, introducing XAI insights at the initialization stage also contributed positively, though less significantly compared to other enhancements. Clearly, the combination of XAI-based chromosome encoding, enhanced initialization, and evaluation in the XAI-Enhanced GA resulted in the best performance among all GA variants.

This section presents convergence plots that track the iterative advancement of different GA configurations over successive generations using three key metrics: evaluation function, transit time, and zone similarity. In these runs, we have eliminated the time constraint on the GA imposed by the competition requirement, and instead set a maximum number of generation equals to 300 to ensure fair comparison. The objectives for these metrics are to minimize transit time, maximize zone similarity, and maximize the evaluation function.

Fig.10 presents the convergence rate of the transit time metric averaged over all testing instances. The dashed red line shows the actual driver's average transit time, providing a real-world desired benchmark. Notice that while the XAI-Enhanced GA converges to a transit time close to the benchmark, the standard GA generates routes that are considerably shorter,

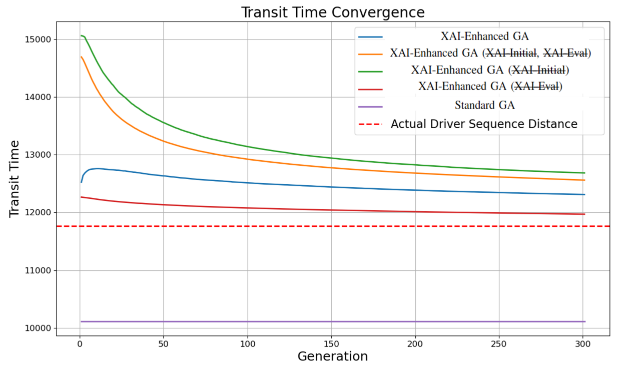

Fig. 10: Transit Time Convergence

suggesting that the driving objective for a desired sequence is not only dependent on the fastest route. Moreover, the standard GA reaches a plateau early in the performance, indicating early convergence to a local optimum; its evolutionary operators prove ineffective in further exploring beyond this local optimum. On the other hand, the *XAI-Enhanced GA* approach shows an initial increase in transit time. This can be explained by the algorithm prioritizing factors like zone similarity, which can temporarily increase transit time. Eventually, the *XAI-Enhanced GA* starts accounting for balancing that with transit time minimization. Notice that the GA variant without the enhanced evaluation components demonstrates superior performance compared to *XAI-Enhanced GA*, likely because it focuses solely on minimizing transit time without incorporating the zone similarity factor.

Fig.11 presents the convergence trends of zone similarity for different GA. The *XAI-Enhanced GA* consistently achieves the highest zone similarity score, demonstrating the effectiveness of SHAP-based evaluation, initialization, and clustering. The GA variants with SHAP-enhanced component significantly improved the zone similarity score from the initial solution. In contrast, the GA variants without the SHAP-informed evaluation showed little to no improvement in zone similarity

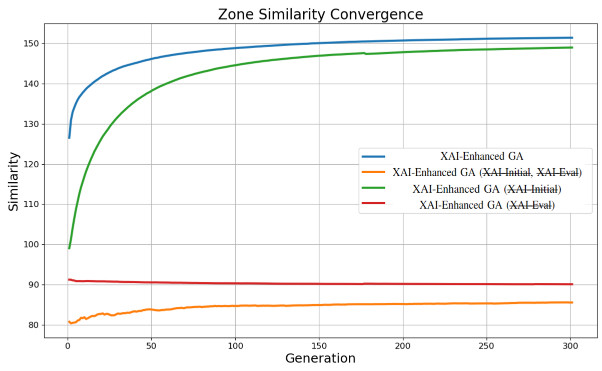

Fig. 11: Zone Similarity Score Convergence

score, indicating the algorithm's struggle to enhance the score without this component. This highlights the limitations of relying solely on traditional optimization evaluation functions to achieve convergence in metrics that are important, but not as obvious such as our zone similarity scores.

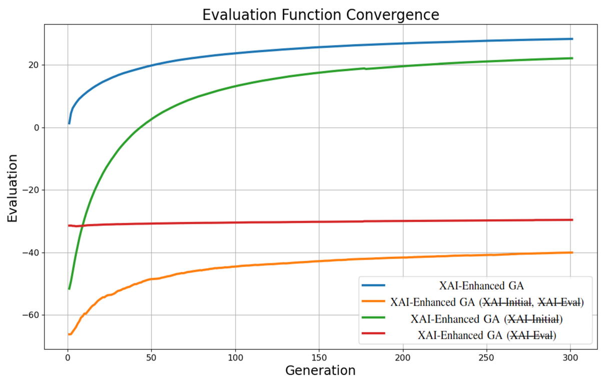

Fig. 12: Evaluation Function Convergence

Fig.12 presents the convergence rates of the fitness function that combines both the transit time and zone similarity score. Similar to Fig.11, the omission of XAI-enhanced evaluation led the algorithm to fail to achieve convergence in the evaluation function, causing the algorithm to stagnate. The two variants with XAI-enhanced evaluation show consistent improvement in convergence across generations. Moreover, XAI-enhanced initialization provides an early advantage for the algorithm, resulting in a better starting point than those without it. This improved initialization could facilitate further evolution of the algorithm. The *XAI-Enhanced GA* that combines XAI-enhanced initialization and evaluation shows the best performance. Their inclusion accelerates convergence and ensures the algorithm consistently finds better solutions.

VII. Conclusions, Limitations, and Future Work

This paper introduces a fusion of ML, GA, and XAI to address a common combinatorial optimization problem. While standard GA is effective in evolving solutions based on given environments, it may not perform optimally when the objective function is unclear or when computational resources are limited. By leveraging ML and XAI, the GA can be equipped with improved understanding of the environment, identifying components beneficial for the evolutionary process.

We test our approach on a route sequencing problem. In the real world, when optimizing delivery routes, drivers often rely on their own insights and experience to adjust planned routes. To overcome this challenge, we propose a two-stage approach using a machine learning phase to learn these insights and experiences, which can then be incorporated into a GA-powered optimization phase. We have illustrated our framework's performance in the Amazon case study.

This research has two limitations that offer opportunities for further improvement and future work. We have implemented and evaluated our framework on one class of combinatorial optimization problems, namely, RSP. This was made possible given the availability of Amazon's Last Mile datasets, which contains human driver behaviors and contextual factors in routing, highlighting the potential importance of feature selection and the need for explainability. Moreover, expanding our work to additional problems would require tailored adaptations, including problem-specific feature engineering and optimization integration. A future research direction includes exploring or curating additional datasets and problems to which this framework can be applied and generalized.

Another limitation is related to the use of SHAP. SHAP interpretations depend on the accuracy of the surrogate model, meaning any biases or inaccuracies may lead to misleading insights. To ensure reliability, it is crucial to verify the input data and validate the surrogate model, as these steps impact the quality of the interpretations. Additionally, SHAP-based analysis can pose scalability challenges due to the computational complexity of calculating feature contributions in large datasets. To mitigate these challenges, model-specific SHAP calculations can be employed to reduce computational costs. For instance, TreeSHAP reduces the complexity to $O(T \cdot L \cdot D^2)$ per instance, where T is the number of trees, L is the number of leaves, and D is the maximum depth of any tree [42]. Alternatively, SHAP analysis and integration can be performed offline prior to the application phase, ensuring that computational demands do not affect the algorithm's execution time in application.

Future research can extend by integrating XAI with other established optimization techniques to enhance the efficiency and applicability of optimization solutions. Additionally, alternative machine learning model constructions could be investigated to handle sequence-dependent data. These advancements would allow for a deeper understanding and application of machine learning with traditional optimization methods.

Appendix A
Complete SHAP Summary Plot

Appendix A shows the complete SHAP summary plot.

Appendix B
LIME Summary Plot

Appendix B shows the LIME plot of top 7 features.

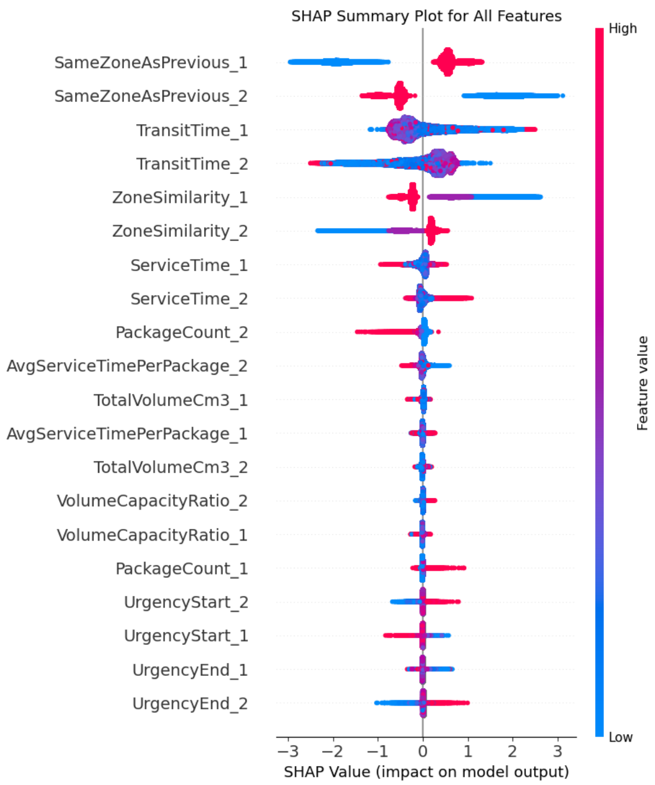

Fig. 13: SHAP summary plot with all features

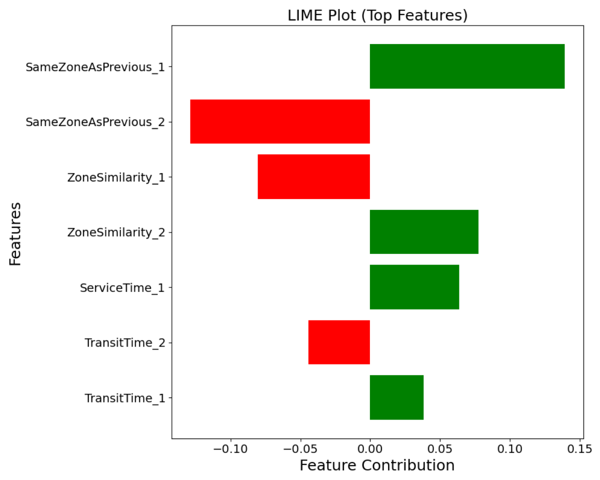

Fig. 14: LIME summary plot with top features

Appendix C
Surrogate Model Experimental Setup

Appendix C presents the hyperparameters evaluated through a grid search using 5-fold cross-validation on the training sets.

TABLE V: Gradient Boosting Grid Search and Accuracy

Parameter	Values Tested / Best
max_depth	[3, 4, 5, **6**]
learning_rate	[0.01, **0.05**, 0.1]
subsample	[**0.8**, 0.9, 1.0]
colsample_bytree	[**0.8**, 0.9, 1.0]
n_estimators	[100, 200, **300**]
Accuracy with Best Parameters: 0.7918	

Appendix D
Genetic Algorithm Experimental Setup

Appendix D lists the Genetic Algorithm parameter settings used for the 3,052 evaluation routes.

TABLE VI: Genetic Algorithm Parameter Setting

Parameter Name	Parameter Value
Population Size	30
Number of Generations	300
Selection Method	Tournament selection (size = 3)
Crossover Rate	1
Mutation Rate	0.4
Number of Elites	3
Elitism Criteria	Transit Time, Similarity, Overall Evaluation (one elite for each criterion)
Evaluation Function Weights	$w_1 = 0.4355$, $w_2 = 0.5645$
Termination Criteria	6.5 seconds

Appendix E
Fitness Function Weights and Results

Appendix E shows results for different evaluation weights.

TABLE VII: Results based on different evaluation weights

Similarity Weight	Travel Weight	Type	Score
0	1	Grid Search	0.0553
0.1	0.9	Grid Search	0.0447
0.3	0.7	Grid Search	0.0415
0.4	0.6	Grid Search	0.0410
0.4355	0.5645	SHAP Weight	0.0411
0.5	0.5	Grid Search	0.0410
0.55	0.45	Grid Search	0.0416
0.7	0.3	Grid Search	0.0430
0.9	0.1	Grid Search	0.0447
1	0	Grid Search	0.0621

References

[1] B. Alhijawi and A. Awajan, "Genetic algorithms: Theory, genetic operators, solutions, and applications," *Evolutionary Intelligence*, vol. 17, no. 3, pp. 1245–1256, 2024.
[2] Y. Jin and B. Sendhoff, "A systems approach to evolutionary multiobjective structural optimization and beyond," *IEEE Computational Intelligence Magazine*, vol. 4, no. 3, pp. 62–76, 2009.
[3] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou, "Metamodel—assisted evolution strategies," in *International Conference on parallel problem solving from nature*, pp. 361–370, Springer, 2002.
[4] Y. Jin, M. Olhofer, B. Sendhoff, *et al.*, "On evolutionary optimization with approximate fitness functions.," in *Gecco*, pp. 786–793, 2000.
[5] S.-H. Wu, Z.-H. Zhan, and J. Zhang, "SAFE: Scale-adaptive fitness evaluation method for expensive optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 478–491, 2021.

[6] C. M. Bishop, "Pattern recognition and machine learning," *Springer google schola*, vol. 2, pp. 1122–1128, 2006.

[7] G. Novakovsky, N. Dexter, M. W. Libbrecht, W. W. Wasserman, and S. Mostafavi, "Obtaining genetics insights from deep learning via explainable artificial intelligence," *Nature Reviews Genetics*, vol. 24, no. 2, pp. 125–137, 2023.

[8] D. Merchán, J. Arora, J. Pachon, K. Konduri, M. Winkenbach, S. Parks, and J. Noszek, "2021 amazon last mile routing research challenge: Data set," *Transportation Science*, vol. 58, no. 1, pp. 8–11, 2024.

[9] D. Dasgupta and Z. Michalewicz, *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.

[10] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: a survey," *Control engineering practice*, vol. 10, no. 11, pp. 1223–1241, 2002.

[11] H. C. Lau, T. Chan, and W. Tsui, "Item-location assignment using fuzzy logic guided genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 765–780, 2008.

[12] X. Zhang, K.-J. Du, Z.-H. Zhan, S. Kwong, T.-L. Gu, and J. Zhang, "Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties," *IEEE transactions on cybernetics*, vol. 50, no. 10, pp. 4454–4468, 2019.

[13] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European journal of operational research*, vol. 167, no. 1, pp. 77–95, 2005.

[14] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary computation*, vol. 4, no. 1, pp. 1–32, 1996.

[15] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia tools and applications*, vol. 80, pp. 8091–8126, 2021.

[16] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pp. 43–55, 2019.

[17] P. Larranaga, C. M. Kuijpers, R. H. Murga, and Y. Yurramendi, "Learning bayesian network structures by searching for the best ordering with genetic algorithms," *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, vol. 26, no. 4, pp. 487–493, 1996.

[18] M. Zbigniew, "Genetic algorithms+ data structures= evolution programs," *Comput Stat*, pp. 372–373, 1996.

[19] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, 2018.

[20] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, "A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 129–142, 2016.

[21] X. Cai, L. Gao, and X. Li, "Efficient generalized surrogate-assisted evolutionary algorithm for high-dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 365–379, 2019.

[22] Q. Lin, X. Wu, L. Ma, J. Li, M. Gong, and C. A. C. Coello, "An ensemble surrogate-based framework for expensive multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 631–645, 2021.

[23] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2009.

[24] R. G. Regis, "Particle swarm with radial basis function surrogates for expensive black-box optimization," *Journal of Computational Science*, vol. 5, no. 1, pp. 12–23, 2014.

[25] K. W. De Bock, K. Coussement, A. De Caigny, R. Słowiński, B. Baesens, R. N. Boute, T.-M. Choi, D. Delen, M. Kraus, S. Lessmann, *et al.*, "Explainable ai for operational research: A defining framework, methods, applications, and a research agenda," *European Journal of Operational Research*, 2023.

[26] K. Coussement and W. Buckinx, "A probability-mapping algorithm for calibrating the posterior probabilities: A direct marketing application," *European Journal of Operational Research*, vol. 214, no. 3, pp. 732–738, 2011.

[27] K. Fleszar, "A MILP model and two heuristics for the bin packing problem with conflicts and item fragmentation," *European Journal of Operational Research*, vol. 303, no. 1, pp. 37–53, 2022.

[28] S. Mitrović, B. Baesens, W. Lemahieu, and J. De Weerdt, "On the operational efficiency of different feature types for telco churn prediction," *European Journal of Operational Research*, vol. 267, no. 3, pp. 1141–1155, 2018.

[29] W. Verbeke, D. Martens, and B. Baesens, "RULEM: A novel heuristic rule learning approach for ordinal classification with monotonicity constraints," *Applied Soft Computing*, vol. 60, pp. 858–873, 2017.

[30] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[31] W. Liu, W. Wei, T.-M. Choi, and X. Yan, "Impacts of leadership on corporate social responsibility management in multi-tier supply chains," *European Journal of Operational Research*, vol. 299, no. 2, pp. 483–496, 2022.

[32] M. De-Arteaga, S. Feuerriegel, and M. Saar-Tsechansky, "Algorithmic fairness in business analytics: Directions for research and practice," *Production and Operations Management*, vol. 31, no. 10, pp. 3749–3770, 2022.

[33] J. Bacardit, A. E. Brownlee, S. Cagnoni, G. Iacca, J. McCall, and D. Walker, "The intersection of evolutionary computation and explainable AI," in *Proceedings of the Genetic and Evolutionary Computation conference companion*, pp. 1757–1762, 2022.

[34] Y.-C. Wang and T. Chen, "Adapted techniques of explainable artificial intelligence for explaining genetic algorithms on the example of job scheduling," *Expert Systems with Applications*, p. 121369, 2023.

[35] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable artificial intelligence by genetic programming: A survey," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 621–641, 2022.

[36] M. J. Walter, D. J. Walker, and M. J. Craven, "An explainable visualisation of the evolutionary search process," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1794–1802, 2022.

[37] M. Fyvie, J. A. McCall, and L. A. Christie, "Towards explainable metaheuristics: PCA for trajectory mining in evolutionary algorithms," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 89–102, Springer, 2021.

[38] S. Corrente, S. Greco, B. Matarazzo, and R. Słowiński, "Explainable interactive evolutionary multiobjective optimization," *Omega*, vol. 122, p. 102925, 2024.

[39] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[40] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[41] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[42] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," *Nature machine intelligence*, vol. 2, no. 1, pp. 56–67, 2020.

[43] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[44] J. Senoner, T. Netland, and S. Feuerriegel, "Using explainable artificial intelligence to improve process quality: Evidence from semiconductor manufacturing," *Management Science*, vol. 68, no. 8, pp. 5704–5723, 2022.

[45] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[46] S. S. Özarık, P. da Costa, and A. M. Florio, "Machine learning for data-driven last-mile delivery optimization," *Transportation Science*, vol. 58, no. 1, pp. 27–44, 2024.

[47] P. Dieter, M. Caron, and G. Schryen, "Integrating driver behavior into last-mile delivery routing: Combining machine learning and optimization in a hybrid decision support framework," *European Journal of Operational Research*, vol. 311, no. 1, pp. 283–300, 2023.

[48] U. Arıkan, T. Kranz, B. C. Sal, S. Schmitt, and J. Witt, "Human-centric parcel delivery at deutsche post with operations research and machine learning," *INFORMS Journal on Applied Analytics*, vol. 53, no. 5, pp. 359–371, 2023.

[49] "Last mile routing challenge: Team performance and leaderboard." Accessed: 2024-05-31.

[50] W. Cook, S. Held, and K. Helsgaun, "Constrained local search for last-mile routing," *Transportation Science*, vol. 58, no. 1, pp. 12–26, 2024.

[51] X. Guo, B. Mo, and Q. Wang, "Amazon last-mile delivery trajectory prediction using hierarchical tsp with customized cost matrix," *arXiv preprint arXiv:2302.02102*, 2023.

[52] O. Arslan and R. Abay, *Data-driven Vehicle Routing in Last Mile Delivery*. Bureau de Montreal, Université de Montreal, 2021.

Figure Captions

Table Captions