Metaheuristic Enhancement with Identified Elite Genes by Machine Learning

Zhenghan Nan, Xiao Wang, and Omar Dib(✉)

Department of Computer Science, Wenzhou-Kean University, Wenzhou, China
{zhenghna,xiaowa,odib}@kean.edu

Abstract. The traveling salesman problem (TSP) is a classic NP-hard problem in combinatorial optimization. Due to its difficulty, heuristic approaches such as hill climbing (HC), variable neighborhood search (VNS), and genetic algorithm (GA) have been applied to solve it as they intelligently explore complex objective space. However, few studies have focused on analyzing the objective space using machine learning to identify elite genes that help designing better optimization approaches. For that, this study aims at extracting knowledge from the objective space using a decision tree model. According to its decision-making basis, a simulated boundary is reproduced to retain elite genes from which any heuristic algorithms can benefit. Those elite genes are then integrated into a traditional VNS, unleashing a remarkable enhanced VNS named genetically modified VNS (GM-VNS). Results show that the performance of GM-VNS surpasses conventional VNS in terms of solutions' quality on various real-world TSP instances.

Keywords: Metaheuristics · Machine learning · Genes selection · Offspring design · NP-hard problems · Elite genes

1 Introduction

Analyzing the structure of the objective space boosts the advancement of metaheuristic optimization simulating natural phenomena as it provides intelligent strategies to explore the search space efficiently. Due to the difficulty of finding a global optimum, metaheuristics have been applied to finding near-optimal solutions for NP-hard problems in a reasonable amount of computational time [11].

Many studies have been conducted to design distinguished algorithms for balancing solutions quality and time complexity. One of the most straightforward approaches is hill climbing (HC), an iterative greedy local search algorithm extensively used in searching problems [12]. However, it has many limitations, especially when the objective space is non-convex; HC tends to get stuck in a local minimum. A simple but effective algorithm is the variable neighborhood search (VNS) designed to handle local minima by systematically exploring distant neighborhood structures [14]. Another metaheuristic is the genetic algorithm (GA), inspired by the natural selection theory. GA has been empirically proven to be an essential candidate for solving large-scale optimization problems in deterministic or stochastic contexts [21]. GAs are robust enough to avoid local minima by using advanced evolutionary strategies such as selection, crossover, and mutations. Both VNS and GA have shown exemplary performance in practice and can evolve with hybridization [7]. Although their application and theory are solid, the research on the influence of distinctive patterns(genes) over metaheuristics is deficient. More specifically, in the process of both GA and VNS, there have been few endeavors to identify significant genes and retain their positive impact on offspring. This paper aims to study whether machine learning models can learn from the structure of an objective space to identify substantial genes as the guidance of metaheuristics enhancement. This paper considers the optimization version of TSP as an example of an NP-hard problem over which the proposed method will be tested. Informally speaking, given a complete graph on n vertices and a weight function defined on the edges, the objective of the TSP is to construct a tour (a circuit that passes through each vertex exactly once) of minimum total weight. The TSP is an example of a hard combinatorial optimization problem; the decision version of the problem is NP-complete. The TSP can be formulated as an integer linear programming problem, as shown in [23].

This study hypothesizes that prominent genes identified from random and local minima solutions are conducive to other metaheuristics on exploration quality. Those unique genes stand out regardless of the host algorithm and lead to better searchability. Firstly, this study generates training data for the decision tree with HC. Secondly, it visualizes the model to study how the decision tree identifies excellent genes. Thirdly, it reproduces the decision-making boundary to guide metaheuristics design based on the tree decision-making. Fourthly, it designs the elite genes conservation generator to insert significant genes into the random solutions as the input of enhanced VNS. Fifthly, it develops the enhanced VNS according to the strong genes' conservation idea. Finally, it compares the traditional and improved VNS to inspire the design of a superior GA.

2 Literature Review

As the fundamental local search metaheuristics, 2-OPT and 3-OPT with the amalgamation of other existing metaheuristics have been widely applied to efficiently tackle NP-hard routing problems such as travelling salesman problem (TSP) and vehicle routing problem (VRP). For example, [2] implemented a hybridization of antlion optimization (ALO) and a 2-OPT algorithm to solve the multi-depot vehicle routing problem (MDVRP) by the extensibility of the 2-OPT algorithm. The results present that the hybridization of metaheuristics outperforms the discrete ALO, GA, and ACO in most cases, proving that the particular extensible attribute is conducive to various traditional metaheuristics. Besides the extensibility, 2-OPT performs well in practice due to its simplicity and fast convergence. For instance, [10] conducted robust research for finding approximate solutions to the TSP, and demonstrated that 2-OPT produces a better solution than christofides' algorithm. However, 2-OPT, as a primary local search, turns out to perform poorly in a complex landscape containing so many local minima. Thus, many researchers focused on either improving other metaheuristics based on 2-OPT or embodying HC strengths into other metaheuristics through hybridization. For example, [6] successfully extracted knowledge from 2-OPT via deep reinforcement learning, promoting a better exploration efficiency. Their study reflects that by exploiting 2-OPT, knowledge about the problem structure can be extracted improving the optimization efficiency. Similarly, this paper researches and takes advantage of the particular compositions of 2-OPT and three similar neighborhood structures, 1-OPT, 3-OPT, and city insertion, as the basis of hybridization in VNS.

With the excellent extensibility of the neighborhood structure, as mentioned before, hill climbing, an iterative local search technique, will empower them in light of finding the local optima solution in the current search region. HC is one of the most general heuristics since it has low operational and time complexity to guide other methods such as evolutionary algorithms efficiently. For example, [16] proposed a novel approach based on HC to control the coupling between the transmitter and a receiver in a telecommunication context. They indicated that HC could produce a much better output than the initial solution. In addition, they argued that HC is cost-effective, giving the system a fast response. In [13], the authors proposed an innovative hybridization between HC and particle swarm optimization (PSO), whose results proved the superiority of hybridization. However, HC can quickly get stuck in local minima, as [1] claimed. Thus, they applied an advanced version of hill climbing named β hill climbing, balancing exploration and exploitation.

This paper proposes a novel elite genes conservation approach based on variable neighborhood search (VNS), as the HC approach tends to get stuck into local optima. VNS algorithm was mainly proposed to handle the local minima issues. VNS switches the neighborhood structure via configurable and flexible shaking and local search functions. For example, in [4], the authors significantly produced better solutions for the wind farm layout optimization problem by combining different shake strategies in VNS, an instructive idea inspiring this paper.

What's more, the flexibility of VNS can prevent the search from falling into local minima, which has been verified by [17] on solving multi-depot green vehicle routing problem. Their result proved that a higher quality solution could be obtained by using multiple neighborhood frameworks to jump out of the basin of a particular search region. Although VNS can effectively handle local minima and combine a variety of neighbor structure search algorithms to find the global optima, there is still a mass of randomness in its search process. Tackling this randomness has resulted in an effective strategy for solving the multiperiod inventory routing problem (IRP), as shown in [8]. The authors used an initial fit solution to avoid unnecessary search iterations and promote VNS performance in their algorithm. Different from their work, this study applies a different technique, namely elite genes conservation, to interiorly dwindle the impact of randomness and intelligently guide the search process.

Both VNS and GA are powerful metaheuristics effectively escaping local minima. Interestingly, the GA imitates a biological evolution process inspired by Darwin's natural evolution theory for its computational model. GA has strong operability since it can directly operate on sets, sequences, matrices, trees, graphs, and other structural objects with different encoding strategies. For example, in [3], the authors studied solving community detection problem (CDP) with GA, and they applied edge-based encoding to represent individuals being evaluated by fitness function straightforwardly. GA is scalable and can be efficiently run in parallel and effectively combined with other algorithms. For instance, [20] conducted a study on combining convolutional neural networks (CNN) with GA for image classification, and they obtained a remarkable improvement in the CIFAR image dataset. In addition, [15] proposed a comprehensive optimization of engine efficiency by coupling artificial neural network (ANN) with GA.

Nonetheless, the mutation in the GA increases the uncertainty of the result. One strategy for improving GA is avoiding blind mutation. [9] adopted that method by proposing a time-varying mutation operator. Compared with traditional GA, using a dynamic mutation operator improves performance. Based on their research, our work points out one of the shortcomings of conventional GA mutation. It puts forward a novel approach to mutating intelligently based on the identified elite genes.

The algorithms above advance significantly with machine learning (ML). Many studies showed that ML has solid performance in extracting knowledge from data. For example, [18] showed that knowledge extraction by ML algorithm promotes the future development of materials science. The ML techniques are widely applied in theory and practice. For instance, [22] argued that combining hybrid metaheuristics, data envelopment analysis, and k-mean clustering is successful for extracting knowledge from metaheuristic algorithms. As a result, extracting knowledge from solutions' structures of metaheuristic algorithms with ML is feasible. This idea has been proved by [20] by classifying genes for early cancer detection and prevention. They reduced the search space and the complexity of the metaheuristic algorithm with ML and data mining.

Moreover, genetic engineering endows individuals with excellent traits by strengthening some genes. For example, [5] mentioned that plant genetic engineering improves crop productivity by giving crops ideal genetic characteristics. Equivalently, identifying elite genes via ML models and maintaining those positive traits to avoid being destroyed during the offspring's optimization will potentially help create strong offspring and lead to faster convergence. Inspired by those ideas, our work applies ML techniques to analyze the structure of objective space, breeding an astute space exploration policy.

3 Methods

This research studies whether a machine learning model can extract useful information from the objective space of TSP, an NP-hard problem in combinatorial optimization. The acquired knowledge from the machine learning model will be used to identifying elite genes with which an intelligent solutions generator and a smart intervention strategy are implemented to enhance existing optimization methods. This work also shows that all the studied algorithms have benefited from the outstanding genes resulting from the ML model; such genes have considerably improved the quality of obtained solutions. As a research methodology, the conventional concepts of VNS and neighborhood structures will be introduced for the convenience of following enhanced algorithms explanation. Secondly, we discuss how the training data of the decision tree model is represented, generated and labeled. Thirdly, the source and representations of instances will be explained. Fourthly, the process of training and visualizing the decision tree will be illustrated. Fifthly, the simulated decision boundary will be reproduced so that the elite genes can be identified. Finally, genetically modified variable neighborhood search (GM-VNS) will be designed based on the elite genes mimicking the simulated decision boundary. In the following, we detail the experimental process.

3.1 Traditional Variable Neighborhood Search (VNS)

X-OPT and city insertion have been extensively studied for the TSP problem. The core idea of 1-OPT is to select an interval for optimization randomly. This algorithm randomly selects one city from the TSP circuit and exchanges it with every other city in the solution to generate the list of neighbor solutions. The process moves to the best neighbor and repeats until a local minimum is encountered. Similar to 1-OPT, 2-OPT and 3-OPT, delete the connection between two and three non-adjacent nodes in the TSP circuit, try other connection modes, calculate the path length after different connection modes, and select the connection mode with the shortest path length as the new solution. Similarly, the city insertion process chooses one element and inserts it into two adjacent cities. Then the process is repeated until a local minimum is encountered. For more information about the adopted neighborhood structures, please refer to [19].

In HC, the local search keeps generating new solutions based on the current solution and comparisons with the current best solution. The latter is updated if the new solution is better than the best. If not, the iteration stops and the best solution is returned. VNS successfully explores a set of predefined neighborhoods to provide a better solution. The algorithm proceeds with an initial solution randomly generated. Next, based on the initial solution and a neighborhood structure, VNS applies a shaking operation that attempts to jump out of the current local minimum region while making the local optima closer to the global optima. After shaking, VNS applies an HC to find the local optima and improve the search accuracy. After HC, the algorithm checks if the resulting solution is improved compared to the best solution. If so, the local minimum will be the new initial solution and search is repeated. If not, the algorithm switches the neighborhood structure and generates a new solution in this neighborhood, and improves it.

3.2 Training Data Generation

As discussed before, this work applies machine learning to analyze the structure of solutions' variables. The data consists therefore of a set of solutions satisfying the requirements of TSP. As for the data features, we use all the edges of a given TSP instance to denote the features. That is, each feature represents an edge, and its value is either 0 or 1. 1 means that the edge is selected to construct the solution (i.e., TSP circuit), whereas 0 means that the edge does not appear in the solution. For example, for a TSP instance with n cities, there exists $(n * (n - 1))/2$ edges |E| corresponding to the number of data features. In order to satisfy the TSP requirements for every data sample, among all the features, only a subset n of |E| will have a value of 1, and the rest of features are set to 0. For the labels, each sample is labeled based upon two values, either 0 or 1. 1 indicates that the sample is a local minimum with relatively higher quality resulting from a local search procedure such as hill climbing, whereas 0 means that the solution is randomly generated, hence of lower quality. The aim is to make two clusters of relatively good and bad solutions, and subsequently analyze the behavior of the decision tree that classifies those solutions. Analyzing the decision boundary helps identifying specific common features in good and bad solutions. Algorithm 1 shows the pseudo-code of the training data generator.

```
Algorithm 1 Trainning Data Generator
Input: trainningDataSize n
Output: List<String>trainningData data
 1: $InstanceSize \leftarrow k$
 2: for $i$ 0 to n by 1 do
 3:     $s1 \leftarrow randomSolutionGenerator(k)$
 4:     $s2 \leftarrow HCwith2OPT(s1)$
 5:     if $s2.Bettterthan(s1)$ then
 6:         $label \leftarrow 1$
 7:     else
 8:         $label \leftarrow 0$
 9:     end if
10:     $newRecord \leftarrow features(edges) + label$
11:     $data \leftarrow data + newRecord$
12: end for
13: return $data$
```

To generate the dataset without bias, a generator randomly selects one city each time and inserts it into the final solution if the selected one isn't repetitive. When all positions of the solution are filled, the generator stops. The random solution of relatively low quality is labeled to 0. As for the local search algorithm used to generate data with label 1, the hill climbing algorithm with 2-OPT, a simple local search algorithm for solving the TSP, is responsible for improving a random solution. There are four neighborhood structures implemented in the experiments, but only 2-OPT is selected as the specification of HC in the data generation phase. The reason is that 2-OPT is known to have the best performance (i.e., produce the best local minima) among the four structures when being applied with HC. An empirical study has been conducted to support that statement, and its results are presented in Table 1.

Table 1. Comparison of HC with different neighborhood structures

Metric\Algorithm	HC-1OPT	HC-2OPT	HC-3OPT	HC-CI
Average distance for TSP (KM)	2501.386	2085.822	2718.354	2422.298

According to Table 1, HC-2OPT has the shortest distance as fitness for the TSP problem compared with other neighborhood structures, namely 2-OPT, 3OPT and city insertion. Therefore, HC-2OPT is selected to generate the relatively higher quality solutions that will have the label 1. We believe that random and higher-quality solutions (i.e., local minima resulting from HC-2OPT) comprise useful information that helps identifying elite genes. This assumption will be verified during the experiments later.

3.3 Test Instances

The distance information of a TSP instance is stored in the form of a matrix specifying the distance value between every pair of cities. Four real-world TSP instances are used in this work: bays29, berlin52, eil76, and eil101. The suffix indicates the number of cities in the instance, so the larger the number is, the more complicated the NP-hard problem will potentially be. Table 2 shows the instances with their descriptions.

Table 2. Description of the selected instances

	Bays29	Berlin52	Eil76	Eil101
Source description	29 Cities in Bavaria (street distance)	52 locations in Berlin (Germany) (Groetschel)	76-city problem (Christofides/Eilon)	101-city problem (Christofides/Eilon)
Optimum tour (km)	2020	7526	538	629

To study the impact of the problem structure, we also use other instances obtained using a simple TSP random generator. The random instance generator can control the instance's range, mean, and variance of distances.

3.4 Visualize the Decision Tree Model to Identify Elite Genes

In the following sections, the decision tree algorithm is applied to the generated datasets to analyze the decision-making criteria and check if it is highly explanatory, facilitating the simulation of decision-making boundaries. The results present that the accuracy of the decision tree model is always very high (>98%) in all the studied instances in this paper. Consequently, it reveals a clear boundary between the local minimum and random solutions. Interestingly, the model has identified some patterns in the data and, therefore, learned from the TSP solution space. Due to substantial decision criteria, the decision visualization is done to study how the decision tree classifies local minimum and random solutions based on several specific edges. According to the visualization, the paths to classify local minima are followed to determine which edges will be fixed and removed. By retaining significant edges, those essential features of outstanding solutions (i.e., local minima) can exert their advantages to optimize future solutions' generators and optimization methods. We refer to those edges as elite genes.

3.5 Genetically Modified Variable Neighborhood Search (GM-VNS)

Retaining outstanding genes (i.e., specific edges in a TSP) by binding two ends (i.e., cities) of edges in VNS is the basic idea of genetically modified variable neighborhood search (GM-VNS). We aim to check whether the identified elite edges can be exploited to improve the performance of a local search procedure such as VNS. The method proceeds with an initial solution generated based on the simulated decision-making resulting from the decision tree classification model. Then locate and package elite genes in the solution in light of simulated decision-making boundary. After the solution is packaged, in the shaking and local search processes, the operations will be based on a set of pre-assembled elite genes rather than single cities as in a traditional VNS. The process is equivalent to manipulating a sequence of pre-identified fit edges instead of manipulating one single city. Following is the flow chart (see Fig. 1) of GM-VNS.

The main difference between traditional VNS and GM-VNS is that GM-VNS operates based on a block of fixed cities rather than single ones. In GM-VNS, the very first

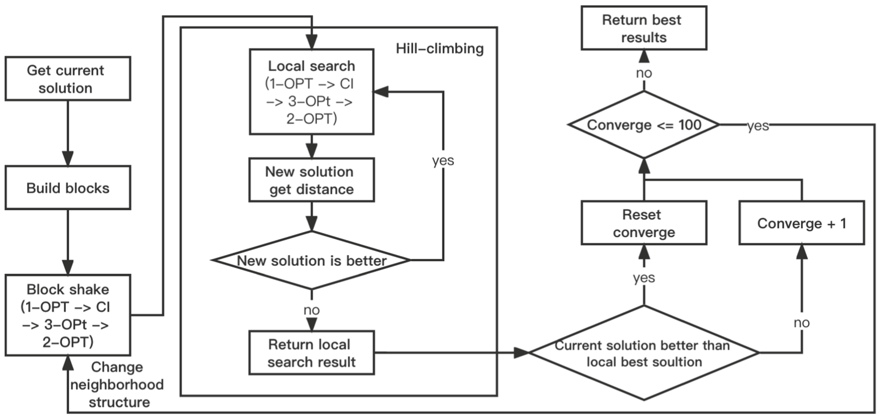
Get current solution
Build blocks
Block shake (1–OPT –> CI –> 3–OPt –> 2–OPT)
Change neighborhood structure
Hill–climbing
Local search (1–OPT –> CI –> 3–OPt –> 2–OPT)
New solution get distance
New solution is better
yes
no
Return local search result
Return best results
no
Converge <= 100
yes
Reset converge
Converge + 1
yes
Current solution better than local best soultion
no

Fig. 1. The flow chart of genetically modified variable neighborhood search (GM-VNS).

operation is block-building. For a TSP problem, traditional local search algorithms use single cities as fundamental elements to change the solution structures. In GM-VNS, a procedure will build blocks of fixed cities before applying local search. That is done by adding the cities of elite edges into an array according to the simulated decision-making basis. A rule list is followed to determine which city should be bound and compute each block's size and elements. It is worth noting that the incumbent solution of GM-VNS already meets the binding requirement by making binding cities adjacent.

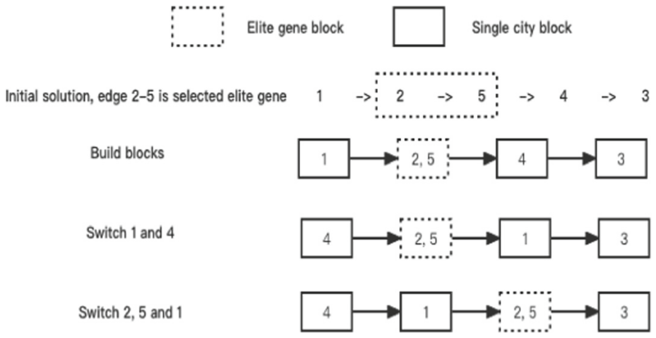

Fig. 2. Illustration of integrating elite genes into VNS

Since the data structure to represent solutions is an array, the block representing solutions is also converted to an array. If the current city should be bound, first, GM-VNS will get its partners and add them to the following block; if not, only the current city will be added to a block with one element. After building blocks, they will be loaded into an array list as a block list. Figure 2 illustrates the concept of blocks and the swap operations underlying it. As can be seen the cities 2 and 5 operates as a single element.

The shaking process in traditional VNS generates a random neighbor solution in the current neighbor structure based on a single element by swapping cities' positions. In

GM-VNS, the swapping operation is based on the blocks of the block list as illustrated in Fig. 2. The aim is to retain some edges based upon the ML decision boundary by binding cities on their sides. After the new shaking strategy, there will be a new neighborhood solution having some edges whose presence will make the solution more fit.

For the local search algorithm, four neighborhood structures are applied based on the blocks of fixed cities and not on single cities. Such algorithms change the positions of certain blocks as in traditional VNS. By doing so, during the HC, every step ensures that the solutions contain those fixed outstanding edges. After HC, a comparison helps update the current best solution. After iterations, the best solution is acquired.

4 Result

4.1 Impact of Dataset Representation

This section studies the impact of data representation on model performance. As mentioned before, the edges represent the features of the labeled datasets in our model. However, one could also use the cities instead of the edges for the features. In that case, the value of each feature represents the order of the city in the TSP circuit.

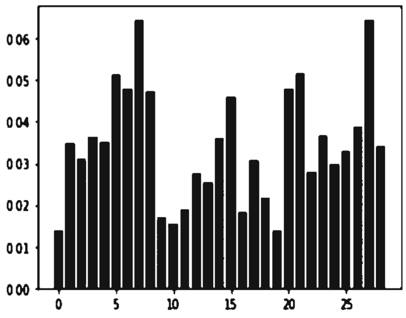

Fig. 3. Feature scores in the decision tree following the "city representation".

Even though the number of features is small, using the city order representation showed that with the instance of 29 cities, after training, the result was a vast tree with too many branches, leaves and paths. The feature scores of each city were similar, meaning that it's difficult to reproduce the simulated decision boundary guiding GM-VNS since each feature has comparable importance. In other words, the city order representation does not help extract knowledge from the objective space effectively. Following are the feature scores (see Fig. 3) in the tree above.

Due to the inefficiency of city order representation, edge representation was adopted. When the features are edges rather than the city index, after training, the tree becomes more insightful (see Fig. 4). The edge representation leads to a more concise tree and has an explicit feature importance difference, meaning that the ML model can effortlessly identify outstanding genes. From a spatial analysis point of view, edge representation outperforms the city order because the objective space constituted seems to be a near

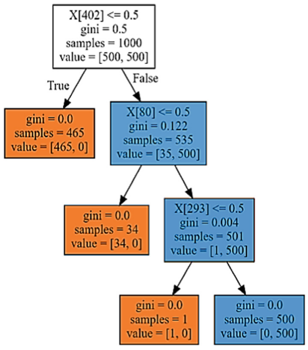

Fig. 4. The visualization of the decision tree following "edge representation".

convex landscape. In contrast, the one constructed by the city order appears to contain many ridges and basins. Naturally, a complex landscape hinders the ML model from identifying decisive factors qualifying the solutions. Hence, the ML model will not generalize well. Interestingly, in the instance with 29 cites, edges 402, 80, and 293 are the most important genes, and they are conserved for the following algorithms.

4.2 Decision Tree Visualization Analysis

Based on the finding that there is a robust decision boundary between local minima and randomly generated solutions, the boundary location is researched in the following sections. Excellent features can be passed on to the next generation by reproducing the boundary to guide solutions' creation. Concerning the edge selection, analyzing instance Bays29 is used to illustrate the process of reproducing decision boundaries.

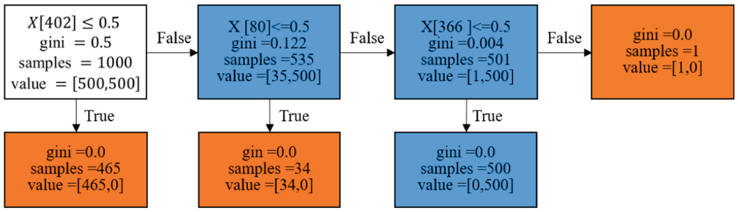

Fig. 5. Decision Tree for Bays29 (Color figure online)

According to Fig. 5, the orange nodes represent random solutions, whereas the blue ones are local minima. In the first node, X indicates the list of features in training data, which corresponds to total edges. It's worth noting that the features' value is either 0 or 1, as introduced before, and the edge is selected only when the value is 1. Hence, $X[402] <= 0.5$ represents a decision on the selection of the edge with index 402. As can be seen, the edge with index 402 is selected to construct the final high-quality solution. Thus, the selected edges inferred from this image are 402 and 80. And the one to be removed is 366. For city nodes binding, the selected edges are converted into a connected city index in terms of the decision.

4.3 Comparison Between Real World and Randomly Generated Instances

Interestingly, with the training data size of 1000, visual analysis shows that the decision trees from random instances are always more complicated (i.e., more leaves, nodes, paths, etc.) than real-world instances, regardless of the degree of randomness involved and instance size. In light of the visualization, the complexity of the tree doesn't vary a lot, even though there is a vital difference among configurations. By contrast, the decision tree of real-world data is always as simple as shown in Fig. 5. In other words, the ML algorithm has more opportunities to identify a set of elite edges. This is an essential characteristic because the more concise the decision-making basis, the more feasible the reproduction of decision boundary will be.

On the contrary, a tree with many leaves and branches disturbs decision-making as each branch might represent a specific local minimum region equivalent to significant genes. The divergence of those genes in the search space triggers the difficulty of integrating their advantages into a single expected solution. That is, a concise tree structure is preferable to keep the eliteness of training data. For real-world instances, the number of clusters containing solutions with outstanding genes is relatively small. It can be described as a flat field with two near basins with unique characteristics that make them shine. That corresponds to a decision tree with few paths (basins) with a particular number of genes. Such a near-convex objective space is conducive to the model's learning. In contrast, the random instances can be seen as a field with so many basins next to each other, which are different and not that deep. That corresponds to a solution space with so many ridges or plateaux. Naturally, learning from such spaces is a difficult task as the existence of many holes with different genes. As such, the ML algorithm will be deficient in generalization ability.

4.4 Impact of the Edges Direction

It's accepted that the orientation of the genes significantly influences their surroundings, affecting the individuals' performance. Similarly, in an NP-hard problem, the direction of solution patterns also affects the solution quality. That's why studying the impact of edges' direction (i.e., genes) is vital for improving traditional metaheuristics.

Table 3. Comparing the impact to solution quality with different bounding directions.

	Metric\Instance	Bays29	Berlin52	Eil76	Eil101
Traditional VNS	Average distance (km)	2053.89	2054.18	2051.44	2052.81
	Average time (ms)	27.404	27.432	27.609	28.196
GM-VNS	Average distance (km)	2046.21	2064.08	2055.42	2068.03
	Average time (ms)	31.914	31.103	32.583	30.94

Like genes, the direction critically influences the surroundings. An experiment was conducted to determine the importance of the edges' direction in solutions. In the instance of 29 cities with 406 edges, according to the simulated decision-making basis, cities "21" and "13", "2" and "28", "16" and "17" are fixed. They were then integrated into GM-VNS to perform the optimization and compared with the solutions' quality of traditional VNS using the same instance but no bound cities. The running time and solution quality of the four testing groups are shown in Table 3.

The comparison results show that the quality of solutions from GM-VNS is different when changing the bounding order in the fixed blocks, proving that the direction of edges impacts the solutions' quality. Likewise, in the biology field, the orientation of adjacent genes has been reported to influence their expression in eukaryotic systems, a convincing analogy fortifying the impact of elite genes orientation on the surroundings.

Therefore, it's essential to find an appropriate bounding direction for adjacent cities, rendering an enhanced performance during the process of exploration and exploitation. Testing the performance in different directions indicates that the best configuration for the 29 cities instance (Bays29) is "21,13", "2,28" and "16,17".

4.5 Conservation Generator Performance Assessment

By applying a simulated decision-making basis, significant genes will be preserved as much as possible as the foundation of the enhanced generator. This subsection aims to study the impact of the quality of the initial solution on the optimization process.

Table 4. Comparison between a traditional and enhanced generator based on elite genes

Metric\Instance	Bays29	Berlin52	Eil76	Eil101
Traditional generator distance (km)	5971.434	29869.411	2501.536	3389.889
Enhanced generator distance (km)	5498.875	28012.23	2384.882	3313.372
Improvement from traditional to enhanced generator in (km)	472.559	1857.181	116.654	76.517
Improvement from VNS to GM-VNS in (km)	13.299	90.469	3.543	2.325

Table 4 presents a substantial enhancement in favor of the assumption that significant genes play an essential role in constructing high-quality solutions. The third row in Table 4 shows that integrating elite genes into the generator results in better initial solutions. By comparing the improvement extent between a traditional VNS with a classical generator and GM-VNS with an improved generator, the fourth row in Table 4 shows that GM-VNS performs better. Therefore, random solutions are easier to be improved than local optima. This phenomenon is apprehensible because the random solutions diverse in objective space contribute to a comprehensive scope searching, but local minimum solutions are usually concentrated in clusters in space. Thus, the impact of outstanding genes diminishes gradually with the solutions' evolution.

Table 5. Comparison of VNS and GM-VNS based on elite genes (average of 200 simulations)

Metric\Instance	Bays29	Berlin52	Eil76	Eil101
VNS distance (km)	2055.998	8018.654	554.597	644.729
GM-VNS distance (km)	2042.699	7928.185	551.054	642.404
Improvement (km)	13.299	90.469	3.543	2.325
VNS convergence time (ms)	37.407	420.523	1826.019	6389.356
GM-VNS converge time (ms)	43.473	535.943	2850.11	8580.196
Time difference	6.066	115.42	1024.091	2190.84

4.6 VNS and GM-VNS Comparison Results

As can be seen from Table 5, GM-VNS always results in better quality solutions than traditional VNS across all four instances. Results demonstrate that retaining outstanding genes did improve the quality of solutions, including local minima. In other words, these originally unobtrusive genes play a significant role, especially when the traditional metaheuristic reaches its performance ceiling. What's more interesting is that the identification of significant genes is based on the local minimum from hill climbing.

However, the experience from HC, a relatively weak algorithm, can enhance VNS performing much better than HC. Therefore, excellent genes are not dependent on a specific algorithm. All the metaheuristics can benefit from them but with different configurations and improvements. The improvement does not necessarily appear to be significant as the increase in instance size. For instance, the improvement of Bays29 is 13.299, and Berlin52 is 90.469, which is much bigger than Bays29. Although Eil76 has a larger instance size than Berlin52, its improvement is 3.543 more minor than Bays29. Therefore, the improvement extent is determined by instance size and other factors such as the instance's distribution and objective landscape. As for the time analysis, results demonstrate that GM-VNS needs a relatively more or similar time to converge compared with VNS. The convergence time is also influenced by the implementation and the additional time to retain and identify dominant genes. As the increase of instance sizes, the time difference becomes more prominent because the NP-hard problems become much more complicated, leading to an increased disadvantage.

4.7 Impact of Elite Genes on Genetic Algorithm

Metaheuristics can perform well with elite genes with different configurations. However, GA might undo the impact of elite genes due to its separate evolution process, which is an interesting direction to improve GA in the future. The defects of traditional GA will be revealed through the following experiment to determine if the strategy of fixing edges in initial solutions enhances the performance of GA. Four groups applied different instances with 29, 52, 76, and 101 cities. For each group, we compare the performance of GA with untreated genes and elite genes. The following are the results of four groups of experiments (average of 200 test data) (see Table 6).

Table 6. Comparison of GA with untreated gene and fixed gene

	Metric\Instance	Bays29	Berlin52	Eil76	Eil101
Traditional GA	Average distance (km)	2080.21	8157.9	594.71	731.40
	Average time (ms)	445.01	1264.92	2227.2	3554.10
Enhanced GA	Average distance (km)	2084.65	8141.28	597.98	736.17
	Average time (ms)	433.87	1290.145	2130.04	3384.18

The difference between GA with untreated genes and fixed genes is slight. In GA, there are processes of blind mutation and crossover, which eventually destroy the fixed edges. Consequently, the result will be similar to the process of using a GA with untreated genes. The results show that the solution using selected genes has lower quality than the solution using untreated genes for the instance Bays29. Nevertheless, the result was reversed in the group using the instance Berlin52. Since the traditional GA mechanisms prevent elite genes from optimizing the internal search process, the quality difference between conventional and improved versions remains negligible. Therefore, the enhanced input solutions injected with elite genes do not benefit traditional GA because the internal evolution process broke and separated those elite genes. Consequently, it's necessary to adapt the genetic operations of the GA to be consistent with the idea of conserving elite genes in the future.

5 Conclusion

Since the conventional VNS and GA neglect specific genes' domination, the systematical evolution process appears indiscriminate. Therefore, it's essential to understand how elite genes guide the evolution of offspring and how they interact with their surroundings. By implementing a genetically modified VNS, this study showed that using the edge representation in the training data is more efficient than the city order representation. The edge representation is beneficial to model learning and generalization.

This study showed that tackling real-world instances differs from random instances from an objective space exploration point of view. Results indicated that real-world instances tend to have a near-convex objective space, whereas the random ones' local optima scatter sporadically in space, impeding the model's learning process. In addition, the genes' direction influence in metaheuristics is predominant, which is persuasive evidence of the interaction between genes and their surrounding environment. Empirical analysis showed that GM-VNS surpasses VNS in solutions quality, verifying those elite genes can strengthen solutions' quality regardless of the algorithm types.

Further research on the genes' environmental interaction and how to take advantage of those elite genes should focus on the following directions. Researchers can develop augmented mutation or mating for GA by dynamically injecting elite genes to avoid blind genetic operators' impact. Other metaheuristics can benefit from elite genes too. Thus, those enhanced metaheuristics are worthwhile to implement. Although the proposed method was only applied to TSP, it is relevant to check its performance on other

NP-hard problems such as knapsack. It's valuable to try different training data representations, contributing to a more effective decision-making boundary. Finally, other machine learning models may reveal more information about elite genes.

References

1. Al-Betar, M.A., Awadallah, M.A., Abu Doush, I., Alsukhni, E., ALkhraisat, H.: A non-convex economic dispatch problem with valve loading effect using a new modified β-Hill climbing local search algorithm. Arab. J. Sci. Eng. **43**(12), 7439–7456 (2018)
2. Barma, P.S., Dutta, J., Mukherjee, A.: A 2-opt guided discrete antlion optimization algorithm for multi-depot vehicle routing problem. Decis. Mak. Appl. Manage. Eng. **2**(2), 112–125 (2019)
3. Bello-Orgaz, G., Salcedo-Sanz, S., Camacho, D.: A multi-objective genetic algorithm for overlapping community detection based on edge encoding. Inf. Sci. **462**, 290–314 (2018)
4. Cazzaro, D., Pisinger, D.: Variable neighborhood search for large offshore wind farm layout optimization. Comput. Oper. Res. **138**, 105588 (2022)
5. Cunningham, F.J., Goh, N.S., Demirer, G.S., Matos, J.L., Landry, M.P.: Nanoparticle-mediated delivery towards advancing plant genetic engineering. Trends Biotechnol. **36**(9), 882–897 (2018)
6. Da Costa, P.R.D.O., Rhuggenaath, J., Zhang, Y., Akcay, A.: Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. arXiv preprint arXiv:2004.01608 (2020)
7. Dib, O., Moalic, L., Manier, M.A., Caminada, A.: An advanced GA–VNS combination for multicriteria route planning in public transit networks. Expert Syst. Appl. **72**, 67–82 (2017)
8. Gruler, A., Panadero, J., de Armas, J., Pérez, J.A.M., Juan, A.A.: A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. Int. Trans. Oper. Res. **27**(1), 314–335 (2020)
9. Hasan, M.M., Kashem, M.A., Islam, M.J., Hossain, M.Z.: A time-varying mutation operator for balancing the exploration and exploitation behaviours of genetic algorithm. In: 2021 3rd International Conference on Electrical & Electronic Engineering (ICEEE), pp. 165–168. IEEE, December 2021
10. Hougardy, S., Zaiser, F., Zhong, X.: The approximation ratio of the 2-Opt Heuristic for the metric Traveling Salesman Problem. Oper. Res. Lett. **48**(4), 401–404 (2020)
11. Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y.: Metaheuristic research: a comprehensive survey. Artif. Intell. Rev. **52**(4), 2191–2233 (2018). https://doi.org/10.1007/s10462-017-9605-z
12. Jately, V., Azzopardi, B., Joshi, J., Sharma, A., Arora, S.: Experimental analysis of hill-climbing MPPT algorithms under low irradiance levels. Renew. Sustain. Energy Rev. **150**, 111467 (2021)
13. Kato, E.R.R., de Aguiar Aranha, G.D., Tsunaki, R.H.: A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing. Comput. Ind. Eng. **125**, 178–189 (2018)
14. Kong, M., Xu, J., Zhang, T., Lu, S., Fang, C., Mladenovic, N.: Energy-efficient rescheduling with time-of-use energy cost: application of variable neighborhood search algorithm. Comput. Ind. Eng. **156**, 107286 (2021)
15. Li, Y., Jia, M., Han, X., Bai, X.S.: Towards a comprehensive optimization of engine efficiency and emissions by coupling artificial neural network (ANN) with genetic algorithm (GA). Energy **225**, 120331 (2021)

16. Rohan, A., Rabah, M., Talha, M., Kim, S.H.: Development of intelligent drone battery charging system based on wireless power transmission using hill climbing algorithm. Appl. Syst. Innov. **1**(4), 44 (2018)
17. Sadati, M.E.H., Çatay, B.: A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem. Transp. Res. Part E Logist. Transp. Rev. **149**, 102293 (2021)
18. Schleder, G.R., Padilha, A.C., Acosta, C.M., Costa, M., Fazzio, A.: From DFT to machine learning: recent approaches to materials science–a review. J. Phys. Mater. **2**(3), 032001 (2019)
19. Sengupta, L., Mariescu-Istodor, R., Fränti, P.: Which local search operator works best for the open-loop TSP? Appl. Sci. **9**(19), 3985 (2019)
20. Sharma, A., Rani, R.: C-HMOSHSSA: gene selection for cancer classification using multi-objective meta-heuristic and machine learning methods. Comput. Methods Programs Biomed. **178**, 219–235 (2019)
21. Sun, Y., Xue, B., Zhang, M., Yen, G.G., Lv, J.: Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE Trans. Cybernet. **50**(9), 3840–3854 (2020)
22. Tayal, A., Solanki, A., Singh, S.P.: Integrated framework for identifying sustainable manufacturing layouts based on big data, machine learning, meta-heuristic, and data envelopment analysis. Sustain. Cities Soc. **62**, 102383 (2020)
23. Rokbani, N., et al.: Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. Soft. Comput. **25**(5), 3775–3794 (2020). https://doi.org/10.1007/s00500-020-05406-5