Self-Attention-Guided Genetic Programming for Dynamic Scheduling: Leveraging BERT for Enhanced Tree-Structured Data Operations

Shanshi Mao , Fangfang Zhang✉ , Yi Mei , Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
{shanshi.mao, fangfang.zhang, yi.mei, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—While Bidirectional Encoder Representations from Trans-formers (BERT) has demonstrated strong performance in learning rich representations from sequential and grid-based inputs like natural language and images, its extension to non-sequential topologies remains an open research question. This study investigates the application of BERT to tree-structured data which presents a significant challenge due to its lack of explicit sequential order and complex topological dependencies. Specifically, we integrate BERT with genetic programming whose classic data representation is tree data structure to solve the dynamic flexible job shop scheduling (DFJSS) problem. DFJSS's inherent computational complexity and highly dynamic, uncertain nature provide a rigorous testbed for our methodology. Our experiments demonstrate that BERT can effectively capture and integrate the structural information embedded in tree-based representations. This finding highlights the versatility and adaptability of the self-attention mechanism, extending its utility beyond conventional sequential or grid-based data structures to a broader class of complex and non-sequential topologies.**

***Index Terms*—BERT, Tree-structured data, Genetic programming, Dynamic scheduling.**

I. Introduction

Dynamic flexible job shop scheduling (DFJSS) is a valuable combinatorial optimization problem with significant practical relevance in manufacturing and processing industries [1]. In DFJSS, a number of jobs need to be processed by a set of machines. Jobs will arrive dynamically, and each job J_j has a sequence of operations $\mathcal{O}_j = \{O_{j1}, O_{j2}, ..., O_{jl_j}\}$ that need to be processed one by one, where l_j is the number of operations of job J_j. Each completed job J_j has a completed time denoted as t_j^{con} and a due time denoted as t_j^{due}. $tard(J_j)$ denotes the tardiness which is calculated as $max\{t_j^{con} - t_j^{due}, 0\}$. Each operation O_{ji} can be processed by more than one machine, and the machine that processes an operation determines its processing time. The objective of DFJSS is to determine effective schedules for processing multiple jobs on a set of machines, where decisions regarding machine assignment and operation sequencing must be made in dynamic environments with continuously arriving jobs [2].

Genetic programming (GP) is a hyper-heuristic algorithm, in which tree-based structures are among the most commonly used representations [3]–[5]. During evolution, GP trees are iteratively modified through operations such as subtree swapping or replacement with newly generated subtrees, introducing substantial flexibility and dynamic variation. This makes the GP a natural fit to learn scheduling heuristics for DFJSS. In the paper, we adopt multi-tree representation where each individual comprises two separate trees, respectively encoding the routing decision for machine assignment and sequencing decision for operation sequencing [6].

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language representation model based on the Transformer architecture [7]. Unlike traditional left-to-right or right-to-left models, BERT leverages bidirectional self-attention to jointly condition on both left and right contexts, enabling it to capture deep contextual dependencies. Recent studies have further extended BERT beyond textual data, demonstrating its effectiveness on non-sequential data such as images. These studies extend the applications of BERT. For example, a recent study [8] employed BERT to encode sequenced tree data, where the tree topology remained flexible while only the terminal nodes are sequenced as input to the encoder. Inspired by this idea, we extend BERT to more flexible and dynamic tree-structured data by considering the entire set of tree nodes, aiming to effectively extract latent information embedded within such flexible representations.

The goal of this paper is to manipulate the GP tree in a more elaborate way by replacing low-contribution subtrees identified through self-attention scores. Our proposed framework is expected to be capable of capturing latent information that can guide effective tree manipulation. To achieve this goal, two key challenges need to be solved:

1. Tree representation for neural models. Unlike fixed-length vectors typically used in neural networks, tree structures are hierarchical and inherently two-dimensional, making direct vectorization challenging. Recent studies struggle to preserve the full complexity of hierarchical relationships, leading to information loss. For example, [9] and [10] representd GP trees using node frequency counts, which neglects the topological relationships within the tree structure.

2. Utilizing BERT's extracted information. Prior studies [11], [12] have shown that multi-head self-attention can capture diverse patterns in sequential (text) and grid-structured (image) data. However, how to effectively interpret and leverage these multi-head outputs remains an open question.

Our proposed framework addresses these challenges and

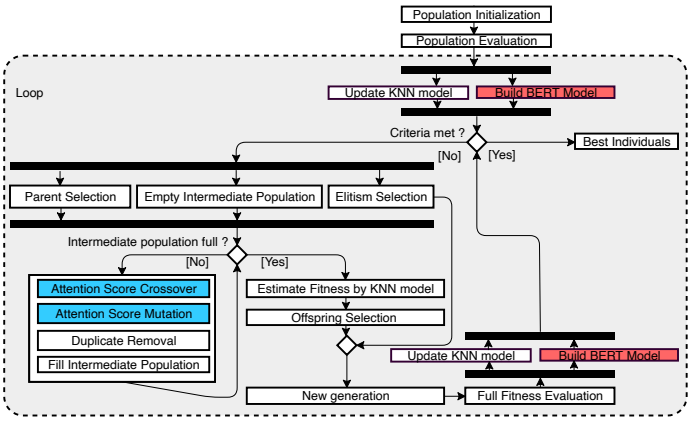

Fig. 1. The flowchart of the proposed algorithm.

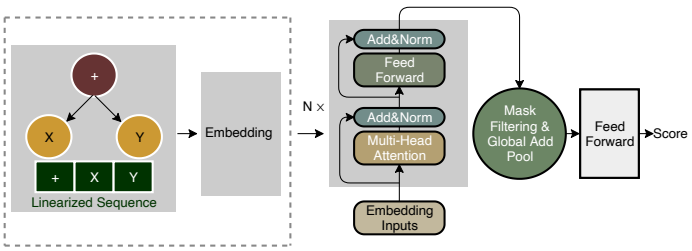

Fig. 2. Neural network architecture for score generation: The input trees are first embedded using the embedding model. The resulting representations are then processed by the BERT encoder, followed by the removal of the padding mask. A global adding pooling operation is applied to obtain fixed-dimensional representations, which are subsequently passed through to produce the final scores.

provides empirical evidence that BERT not only learns meaningful structural patterns in tree-based data but also enables fine-grained tree manipulation through self-attention-guided subtree replacement. The core assumption underlying our approach is that different heads capture different patterns, and we aggregate node-level self-attention scores across all heads. Nodes consistently receiving low scores across all heads are treated as trivial, allowing their corresponding subtrees to be intelligently replaced with newly generated or more meaningful structures. The main contributions of this paper are summarized as follows:

1. Tree data vector representation: We propose a novel representation that captures the hierarchical and topological relationships within tree structure, avoiding the information loss inherent in conventional methods.

2. Attention-guided genetic operators: We develop genetic operators guided by self-attention scores extracted from BERT. By exploiting the model's ability to identify significant and trivial substructures, our approach enables a more informed and effective search.

3. Extending attention mechanisms to complex topological data: We empirically demonstrate that attention mechanisms can effectively extract structural features and dependencies from highly irregular, tree-structured data. This validates their utility in domains beyond sequential or spatial data.

II. Proposed Algorithm

A. An Overview

A population of N individuals is first produced as the initial generation and is subjected to a full evaluation. This evaluated population then forms the training dataset for the BERT model, as highlighted by the red box in Figure 1. The evolutionary process proceeds in iterations. At the beginning of each iteration, a check is performed to determine if a predefined termination criterion has been met. If so, the process terminates, and the best individual from the last generation is outputted. Otherwise, the iteration continues. The previous generation undergoes GP operators for selection, self-attention score based mutation and crossover(highlighted by blue box) to produce a new intermediate population. Rather than randomly selecting the subtrees, our framework identifies the node with the lowest self-attention score and treats the subtree rooted at this node as the target for modification. Specifically, mutation replaces this subtree with a randomly generated subtree, while crossover replaces it with the subtree having the highest self-attention score from another individual. More details are provided in Section II-F. Once the intermediate population is generated, a surrogate model is used to preselect the most promising individuals for the next generation, this process follows Hildebrandt and Branke's method [13].

After the new generation is composed, its individuals are fully evaluated to acquire their true fitness values, which are then used to update the surrogate and BERT model. To enhance model generalization, a comprehensive global dataset is maintained for BERT training. Throughout the evolutionary process, individuals from each generation are progressively integrated into this dataset, thereby augmenting the training pool and enriching the diversity of the data samples. Finally, if the termination criterion has not been met, a new iteration begins. During the iteration, a new BERT model is built in each iteration rather than updating the existing one for the reason: The genotypic makeup of individuals tends to converge as the iterations progress. As a result, the training parameters from earlier generations become less relevant and can negatively impact the BERT model's performance on the new, more converged population.

B. Proposed Architecture

As shown in Figure 2, the proposed architecture consists of three main components: 1) the embedding (embraced by dashed box) 2) the BERT encoder 3) the global pooling and feed forward neural network. The mathematical definition is denoted as:

$$Score = GlobalAddPooling(BERT(Embeeding(\phi))) \cdot W^T \quad (1)$$

Pooling is employed to standardize the dimensionality of the representations, ensuring that the input to the feed-forward neural network remains consistent regardless of the initial sequence length.

1) Linearize and Embed: As shown in the formula (1), we denote ϕ as the linearized padded tree representation. By

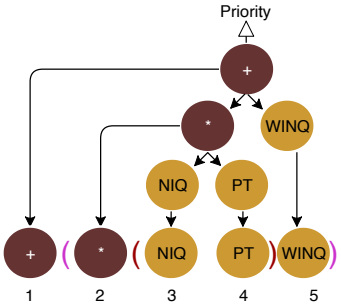

Fig. 3. The two-dimensional GP tree can be linearized into a one-dimensional sequence using DFS, resulting in a Polish notation forlmula whose semantic information can be interpreted as a text sequence.

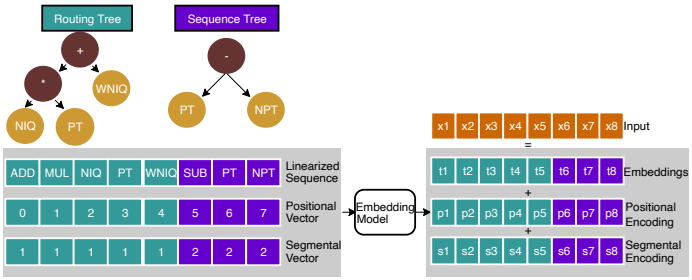

Fig. 4. The process of forming a sequence from two trees and embedding this sequence as the input. This corresponds to the detailed steps within the dashed box of Figure 2.

depth-first search (DFS), a GP tree would be linearized as a sequence, and we combine the sequence tree and routing tree of single individual into one sequence by concatenating them into one sequence. To distinguish the different linearized sequences, the segmental vector is added to indicate the different trees. The learned embedding model is updated for training BERT model.

2) BERT Encoder: The embedded sequence is encoded by BERT to extract the latent features by utilizing the self-attention mechanism. Since different heads capture different patterns within the sequence, we assume that a node with consistently low self-attention scores across all heads is less relevant to the other nodes. Based on this assumption, we propose a self-attention-score–based mutation and crossover mechanism. More details are provided in Section II-F.

3) Feed Forward Neural Network: To assess an individual from semantic aspect, after the encoding, we should integrate the information extracted from encoder. To achieve this, the global adding pooling method is used. The detail is, the encoder output is firstly removed out the padding sequences, and by adding all node vectors to implement the adding pooling process. The mathematical formula is shown as $\mathbf{r} = \sum_{n=1}^{N_i} \mathbf{x}_n$, where the $\mathbf{x}_n$ represents the n-th encoded token vector, and $\mathbf{r}$ is the adding pooling output vector. Finally, each individual is processed into a vector as the input of the multi-layer of feed forward neural network (FFN). The FFN will output the score which is used to evaluate the individuals. The reasons for employing a score, rather than a direct fitness value, are discussed in Section II-E.

We define two distinct scoring categories: (1) Individual Score, a fitness-related metric reflecting the loss value (discussed in Section II-E), and (2) Node Score, specifically the self-attention scores generated by the BERT architecture. To maintain clarity, score will refer to the overall evaluation of a GP individual, while self-attention score signifies the internal self-attention score assigned to each node.

C. The Semantic Information in Tree-based GP Individual

A key fact is that a tree-based GP individual possesses an inherent semantic meaning. However, extracting this semantic content directly from its tree representation is a significant challenge. To capture this semantic information, we propose to transform the GP tree into a text sequence, as shown in Figure 3. Specifically, by using DFS traversal, the GP tree in Figure 3 can be converted into the Polish notation format [14], such as $+(*\ (NIQ, PT), WINQ)$. Further, we can unnest this expression into a simplified text sequence like $+, *, NIQ, PT, WINQ$. Finally, the linearized sequence is readable, similar to a standard sentence. Therefore, we suppose that this linearized tree-based GP representation contains valuable semantic information, which inspires us to encode it as a sequence. Linearizing GP tree offers several key advantages:

1) Comprehensive Representation: Unlike other methods, such as the frequency representation [9], [10], which consider only node frequencies and ignore topology, our sequence representation incorporates all nodes while the linearization mechanism preserves their topological information. This comprehensive approach ensures that critical performance-related information is retained.

2) Compatibility with Advanced Encoders: Text sequences are easy to manipulate and are well-suited for state-of-the-art models like the Transformer. Recent studies [12], [15] have demonstrated that the multi-head self-attention mechanism within the Transformer architecture is exceptionally good at encoding informative representations from sequential data.

In summary, by linearizing the GP tree, we leverage these advantages to create a representation that is not only comprehensive in its inclusion of all node information but is also highly amenable to informative encoding methods.

D. Embedding Linearized GP Sequence

In this paper, we use two separate trees to represent the sequencing rule and the routing rule. To linearize these two trees into a single sequence, we concatenate them and use a segmental vector to identify each part. The embedding schema for this process is shown in Figure 4 where each input vector is defined as $x_i = s_i + p_i + t_i$, with s_i, p_i, and t_i denoting the i-th segmental encoding, positional encoding, and sequence embedding vectors, respectively.

E. The Score and Pair-wise Loss Function

In many studies [9], [10], [13], the learned models are used to predict fitness values directly. However, direct fitness prediction suffers from a data augmentation limitation due to the rotation of training instance at each generation [16]. To address

Algorithm 1 Pairwise Loss Function

Input: scores, labels, λ_{var}
Output: loss value $\mathbb{L}$

$\mathbb{L} \leftarrow 0$
variation $\leftarrow 0$
$n \leftarrow$ length of labels
mask $\leftarrow$ list of n zeros
unique score list $\leftarrow []$
groups $\leftarrow$ unique elements in labels
for each $g \in$ groups **do**
    indices $\leftarrow \{i \mid label_i = g\}$
    values $\leftarrow \{score_i \mid i \in \text{indices}\}$
    $\mathbb{L} \leftarrow \mathbb{L} +$ Variance(values)
end for
$\mathbb{L} \leftarrow \mathbb{L} \cdot \lambda_{\text{var}}$
for $i = 0 \ldots n-1$ **do**
    for $j = i \ldots n-1$ **do**
        if labels$[i]$ = labels$[j]$ **then**
            $\mathbb{L} \leftarrow \mathbb{L} + |\text{scores}[i] - \text{scores}[j]|$
        else
            if labels$[i]$ > labels$[j]$ **then**
                sign $\leftarrow -1$
            else
                sign $\leftarrow 0$
            end if
            $\mathbb{L} \leftarrow \text{ReLU}\big(\mathbb{L} + (\text{scores}[i] - \text{scores}[j]) \cdot \text{sign}\big)$
        end if
    end for
end for
return $\mathbb{L}/(n \cdot (n-1)/2) +$ variance

these challenges, we propose training the learned model to output a score for each individual instead of predicting its fitness value directly. The primary objective is to align the rank of individuals based on these scores with the rank based on their true fitness values.

This paper proposes a pair-wise ranking loss function as shown in Algorithm 1. It begins by grouping individuals based on their fitness values. As the evolutionary process progresses and the population converges, individuals often achieve identical fitness scores. These individuals are partitioned into distinct groups, and we subsequently compute the score variance within each group. The underlying principle is that a well-calibrated predictive model should assign scores with minimal variance to individuals that share the same fitness value. We hypothesize that for individuals exhibiting identical fitness values, the variance in their individual scores should remain minimal, reflecting a high degree of performance consistency. This approach provides a measure of the model's consistency and predictive reliability for equally-performing individuals.

Following the variance-based analysis, we employ a pair-wise ranking loss function to compute the overall loss value. The core premise of this function is to enforce a consistent relationship between the predicted scores and the actual fitness ranks. If two individuals possess the same fitness value, the loss function is designed to penalize any discrepancy in their predicted scores. Conversely, if there is a rank mismatch—where the predicted score rank does not align with the true fitness-based rank—the difference between their scores is added to the total loss. This mechanism directly guides the model to learn the correct ordinal relationships among individuals. To ensure that only ranking errors contribute to the loss, we apply the Rectified Linear Unit (ReLU) function to the computed loss values. This function ensures that if the predicted rank is correct (i.e., the pairwise difference is non-positive), the resulting loss is zero. If the rank is incorrect, the loss value remains positive, thereby penalizing the model for misordering.

TABLE I
THE PARAMETER SETTINGS OF THE PROPOSED METHOD.

PARAMETER	VALUE	PARAMETER	VALUE
Population size	50	Elitism	10
Generations	50	Maximal depth	8
Instances/generation	2	Crossover rate	0.80
Initialisation method	Ramped-half-half	Mutation rate	0.15
Min / max depth	1 / 6	Reproduction rate	0.05
Parent selection	tournament (3)	Self-attention	0 / 0.8

F. Score-based Crossover and Mutation

In GP, crossover and mutation are used to introduce randomness and encourage exploration and exploitation. While classical GP applies these operations to random subtrees, our approach uses the self-attention mechanism to evaluate subtrees at the node level, providing a more informed guidance.

The different heads within a BERT model are designed to capture various patterns in sequential data, such as focusing on specific token types, positional relationships, or the entire sequence [11]. In our work, we use 8 attention heads to capture the patterns within the linearized Polish notation formulas. Each head assigns a score to every node, indicating how much attention it receives from other nodes under that specific pattern. We then sum these 8 scores to get a total score for each node, which represents its overall importance: $N_{\text{score}} = \sum_{i=1}^{8} \text{Score}_i$, where N_{score} is the total score for a node and Score_i is the attention score from the i-th head.

Our core assumption is that a node with a very low total self-attention score is less important in all learned patterns. In our approach, since both the routing tree and the sequence tree are flattened into a single token sequence, the eight attention heads can attend to tokens across both trees. This design is intuitively reasonable because the fitness value is determined jointly by the both trees. Based on the self-attention score assigned to each node, this allows us to make more targeted genetic operations. Instead of random replacement, we can replace the subtree rooted at a low-scoring node with a subtree from another individual that has a high-scoring node (crossover).

III. RESULTS AND DISCUSSIONS

This paper targets on minimizing the tardiness. The experimental results are statistically validated using Friedman's test and the Wilcoxon rank-sum test with a significance level of 0.05. Each individual of the algorithm is composed of terminals and a set of functions, as suggested in [17]. Table I provides details of the hyper-parameters. To analyze the effect

TABLE II
THE MEAN (STANDARD DEVIATION) OF OBJECTIVE VALUES ON TEST INSTANCES OF GP, KNN-GP AND PROPOSED BERT-SSGP.

Algorithms	HH	HL	LH	LL	Rank
GP	830.94(146.86)(+)	345.61(98.47)(+)	1846.50(331.78)(+)	779.18(204.86)(≈)	3.0
KNN-GP	794.20(136.84)(+)	306.43(52.92)(+)	1684.41(127.83)(≈)	698.45(71.62)(≈)	1.8
BERT-SSGP	**728.91(98.57)**	**290.27(57.13)**	**1661.72(150.32)**	725.13(109.05)	1.2

(+), (−), and (≈) denote that BERT-SSGP is significantly better, worse, or similar to algorithms.

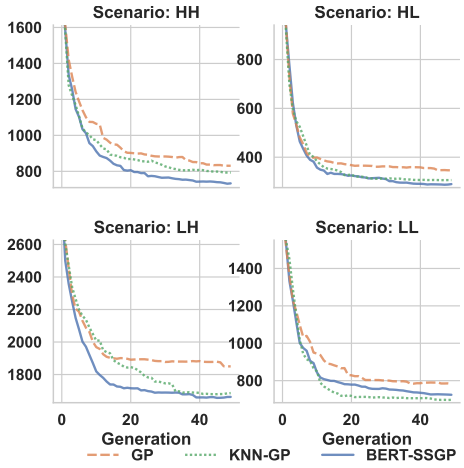

Fig. 5. The curves of average fitness values according to 30 independent runs on test instances of GP, KNN-GP, BERT-SSGP with 50 generations.

of attention score-based mutation and crossover, we set up a self-attention score utilization of 0% and 80%. The 80% self-attention score utilization means 80% probability to take self-attention-score-based mutation and crossover as described in Section II-F. This parameter also control the balance between exploration and exploitation. In this paper, we use the dataset suggested in [18].

A. Overall Performance of Proposed Algorithm

In the following discussion, we denote the baseline model as KNN-GP which does not utilize self-attention scores to guide the genetic operators (mutation and crossover) [13]. The proposed model, referred to as BERT-SSGP (self-attention-score-based GP), extends the KNN-GP model by incorporating self-attention scores as guidance for the genetic operators. Specifically, in this model, there is an 80% probability that mutation and crossover are performed under the guidance of self-attention scores. KNN-GP is a highly effective and easily implementable GP framework that remains a widely adopted baseline in recent works [10], [17], [19]. Consequently, KNN-GP serves as an ideal baseline to validate the efficacy of leveraging BERT for processing tree-structured data.

Firstly, as shown in Figure 5, in both HH and LH scenarios, BERT-SSGP achieves a faster convergence rate than the baseline model. This indicates that, compared with random subtree selection, self-attention score based subtree selection is more effective as the self-attention scores provide informative signals that help identify inferior subtrees.

Secondly, according to Table II, BERT-SSGP achieves the highest overall ranking among all models and outperforms the baseline model in two out of the four scenarios. However, in LH and LL scenarios, BERT-SSGP does not outperform the baseline because of the limitation that self-attention score mechanism could only increase the probability that the new tree generated by self-attention score guided genetic operations is better, but it can not guarantee improvement in every case.

The results support the core hypothesis that a more guided, less random approach to mutation and crossover operations leads to more effective and efficient search processes within a GP framework.

B. Does Attention Score Can Really Guide the Evolution?

We propose a hypothesis to explain this: by targeting and replacing nodes with the lowest average self-attention scores, we can potentially increase the lower bound of an individual's overall score. While this mechanism does not guarantee an improvement with every operation (because new subtrees are generated randomly), it increases the probability of improvement. Let's represent this process to understand the potential for improving an individual's performance.

This paper represents the entire tree structure as T, which consists of N nodes, $\{v_1, v_2, \ldots, v_N\}$. Each node v_i has an average self-attention score, $s(v_i)$. The overall performance of the tree, $E(T)$, is defined by its minimum node self-attention score as $E(T) = \min_{i=1}^{N} s(v_i)$. Let $v_{\min}$ be the node with the lowest self-attention score in tree T, so $v_{\min} = \arg\min(s(v_i)), i \in \{1, 2, \ldots, n\}$, and its minimum self-attention score is $s_{\min} = s(v_{\min})$. In a genetic operation, we replace the subtree rooted at $v_{\min}$ with a new, randomly generated subtree, T_{new}. This new subtree contains m nodes, $\{u_1, u_2, \ldots, u_m\}$, each with a self-attention score $s(u_j), j \in \{1, 2, \ldots, m\}$. The new tree, T'(the new tree after the replacement), has a minimum score of:

$$E(T') = \min(\{s(v_i) \mid v_i \in T, v_i \neq v_{\min}\} \cup \{s(u_j) \mid u_j \in T_{new}\})$$

If all node self-attention scores in the new subtree, T_{new}, are greater than or equal to the original minimum self-attention score, $s(u_j) \geq s_{\min}$ for all j, then the new tree's minimum score will be at least as good as the original, i.e., $E(T') \geq s_{\min}$. This is the ideal scenario where the operation is guaranteed to improve or maintain the tree's lowest self-attention score. However, since the new subtree is generated randomly, it might contain nodes with self-attention scores lower than the original minimum. In this case, the new tree's minimum self-attention score, $E(T')$, could be lower than $E(T)$. But by targeting the original lowest self-attention scoring node for replacement, we are increasing the probability of generating a new subtree with a higher minimum self-attention score, thereby raising the overall lower bound of the individual.

C. How Does BERT Assess Self-Attention Score to Nodes?

Figure 6 shows 3 heads attention wights among the input sequence. The experimental results reveal three primary patterns of attention:

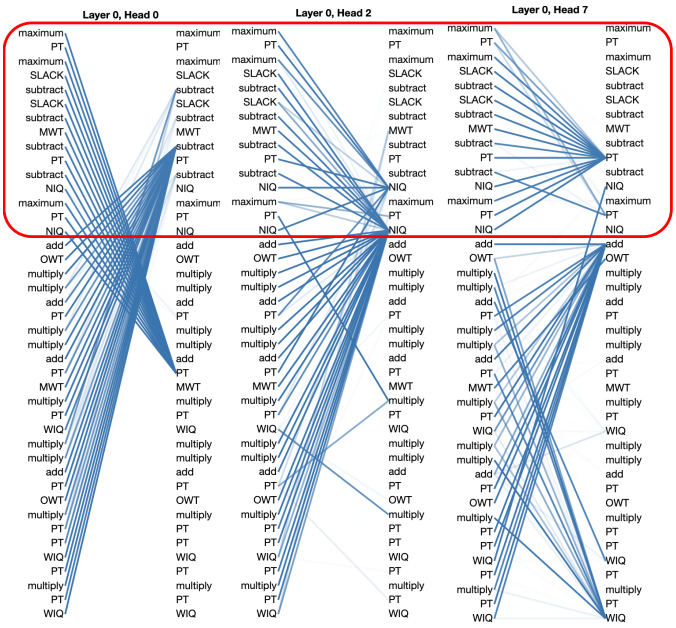

Fig. 6. BERT attention heads, i.e., in the example attention maps, the darkness of a line indicates the strength of the attention weight.

Inter-Sequence Attention: Certain attention heads, such as Head 0, predominantly attend to tokens from both sequences. This behavior suggests the model is learning the vital dependencies and interactions between the two rules, which is crucial given their collective influence on the final outcome.

Intra-Sequence Attention: In contrast, other heads, including Head 7, tend to focus their attention primarily within their own sequence. This indicates that these heads are specializing in capturing the unique, inherent features and hierarchical structure of each individual rule.

Integrated Attention: A third group of heads, exemplified by Head 2, demonstrates a more balanced, integrated approach. These heads attend to both their own sequence and the other sequence simultaneously, suggesting a holistic understanding of how local features and broader inter-sequence relationships contribute to the overall representation.

This diverse range of attention patterns implies that BERT is capable of effectively identifying and learning the most critical information—both internal to each rule and in the interactions between them within the combined tree-based sequences.

IV. Conclusions and Future Work

Our findings reveal that self-attention scores hold significant meaning within the tree structure of GP individuals. By leveraging these scores to guide genetic operations like mutation and crossover, we can effectively elaborate individuals at the genotypic level. This advancement broadens the methodological scope of GP evolution. Furthermore, our experimental results demonstrate the efficiency of the self-attention mechanism on tree structures. We show that BERT can effectively extract both the tree's topology and the latent information embedded within its nodes.

However, our proposed approach suffers from a significant increase in training time due to attention score computation complexity which is O(n2), where n is the sequence length. We will investigate this in the near future.

References

[1] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: An evolutionary learning approach," in *Machine Learning: Foundations, Methodologies, and Applications*. Springer, 2021, pp. XXXIII+338.

[2] K. Jaklinović, M. Durasević, and D. Jakobović, "Designing dispatching rules with genetic programming for the unrelated machines environment with constraints," *Expert Systems with Applications*, p. 114548, 2021.

[3] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[4] L. Planinić, M. DJurasević, and D. Jakobović, "On the application of ϵ-lexicase selection in the generation of dispatching rules," in *Proceedings of the Congress on Evolutionary Computation*. IEEE, 2021, pp. 2125–2132.

[5] N. Pillay and R. Qu, *Hyper-heuristics: theory and applications*. Springer, 2018.

[6] F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 472–484.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics)*, Jun. 2019, pp. 4171–4186.

[8] M. C. Teixeira and G. L. Pappa, *Transformers as Surrogate Models for Genetic Programming in AutoML Tasks*. New York, NY, USA: Association for Computing Machinery, 2025, p. 472–480.

[9] L. Tan, C. Jin, X. Chen, R. Qu, and R. Bai, "Pgu-sgp: A pheno-geno unified surrogate genetic programming for real-life container terminal truck scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2025, pp. 322–330.

[10] L. Zhu, F. Zhang, X. Zhu, K. Chen, and M. Zhang, "Phenotype and Genotype Based Sample Aware Surrogate-Assisted Genetic Programming in Dynamic Flexible Job Shop Scheduling," *IEEE Transactions on Artificial Intelligence*, pp. 1–15, 2025.

[11] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does bert look at? an analysis of bert's attention," in *BlackboxNLP@ACL*, 2019.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *ArXiv*, vol. abs/2010.11929, 2020.

[13] T. Hildebrandt and J. Branke, "On using surrogates with genetic programming," *Evolutionary Computation*, vol. 23, no. 3, pp. 343–367, 2015.

[14] Wikipedia contributors, "Polish notation — Wikipedia, the free encyclopedia," 2025, [Online; accessed 20-July-2025].

[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV: European Conference, Glasgow, UK, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 213–229.

[16] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance-rotation-based surrogate in genetic programming with brood recombination for dynamic job-shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1192–1206, 2022.

[17] ——, "Instance-Rotation-Based Surrogate in Genetic Programming With Brood Recombination for Dynamic Job-Shop Scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1192–1206, Oct. 2023.

[18] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Niching genetic programming to learn actions for deep reinforcement learning in dynamic flexible scheduling," *IEEE Transactions on Evolutionary Computation*, 2024.

[19] R. Chen, Y. Mei, F. Zhang, and M. Zhang, "Neural network surrogate based on binary classification for assisting genetic programming in searching scheduling heuristic," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2024, pp. 309–321.