Fitness Landscape Optimization Makes Stochastic Symbolic Search By Genetic Programming Easier

Zhixing Huang [1], Yi Mei [1], *Senior Member, IEEE,*
Fangfang Zhang [1], *Member, IEEE,* Mengjie Zhang [1], *Fellow, IEEE*
Wolfgang Banzhaf [2], *Member, IEEE*

***Abstract*—Searching for symbolic models plays an important role in a wide range of domains such as neural architecture search and automatic program synthesis. Genetic programming is a promising stochastic method for searching effective symbolic models within an acceptable time. The genetic programming performance is closely related to the hardness of the fitness landscape. A better fitness landscape with less local optima normally implies that it is easier to search for better solutions. In recent years, there have been many studies enhancing genetic programming performance by forming better fitness landscapes. However, the better design of the fitness landscape highly relies on specific domain knowledge and consumes a lot of expert effort. This paper proposes a fitness landscape optimization method to automatically design better fitness landscapes for genetic programming search than the manually designed ones. We optimize the landscapes by optimizing the neighborhood structures of symbolic solutions. We verify the effectiveness of the proposed method in both supervised learning and combinatorial optimization problems. The results show that the proposed method significantly reduces the hardness of fitness landscapes. By simply searching against the automatically optimized fitness landscapes, a genetic programming method can have a very competitive performance with state-of-the-art methods.**

***Index Terms*—Fitness landscape, neighborhood structure, stochastic symbolic search, genetic programming.**

I. Introduction

Searching for symbolic solutions is the core optimization process in many artificial intelligence tasks such as neural architecture search [1], digital circuit design [2], and program synthesis [3]. Searching for symbolic solutions is vastly different from searching for numerical solutions. For example, there is a set of syntax rules for defining symbol combinations or topological structures, which leads to irregular search spaces. The physical meanings of similar symbol solutions are greatly different from each other, which leads to a weak causality between symbolic solutions and their behaviors. These distinctive features make symbolic search a challenging task. In practice, the search space for symbolic solutions is tremendously large, and we often lack a clear insight into effective symbolic solutions. Therefore, we have to stochastically search for possible solutions, hoping to find effective feasible ones in a reasonable time, that is, stochastic symbolic search.

Each stochastic symbolic search problem has a fitness landscape (FL). An FL is a surface that reflects the fitness of all the possible solutions in a search space [4]. An FL with less local optima (e.g., an unimodal landscape) normally implies an easier symbolic search problem. For example, with a less-local-optima FL, a simple iterative local search can outperform well-designed metaheuristic methods in some neural architecture search problems [5]. An FL consists of three components: fitness function, solution space, and the neighborhood structure of solutions [6]. A fitness function measures the effectiveness and quality of all the possible solutions, the possible solutions constitute the solution space, and the neighborhood structure defines the neighbors of each solution. However, the FLs of symbolic search problems are normally extremely rugged because of the low causality among symbolic solutions (i.e., a small change in a computer program might lead to a huge change in the final output). Rugged FLs make symbolic search very challenging.

Genetic programming (GP) is a typical evolutionary computation method that searches for symbolic solutions [7]. GP evolves a population of variable-length individuals. Each individual represents a symbolic solution such as a computer program or a mathematical formula. GP directly searches symbol combinations and their topological structures, which essentially define the search space and neighborhood structures of FLs. Applying GP to search symbolic solutions has attracted many research interests [8], [9] and has been extended to various applications, such as image classification [10], [11], regression [12], [13], and dynamic combinatorial optimization problems [14].

In recent years, some advanced techniques have enhanced GP performance in searching symbolic solutions by essentially designing better FLs. For example, multitask GP [15] helps GP jump out from local optima by cooperating with similar landscapes. Feature selection [16] and frequency-based operators [17] change the neighborhood structures (e.g., one-hop mutation) so that GP prefers particular neighbors with a large number of certain features. However, these manually enhanced fitness landscapes need very specific domain knowledge and strong assumptions. For example, in multitask GP, one has to find two (or more) correlated tasks whose fitness landscapes are synergic. In frequency-based mutation, users

The authors are with [1] the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (E-mail: zhixing.huang@ecs.vuw.ac.nz; yi.mei@ecs.vuw.ac.nz; fangfang.zhang@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz), and [2]Department of Computer Science and Engineering, BEACON Center for the Study of Evolution in Action, and Ecology, Evolution and Behavior Program, Michigan State University, East Lansing, MI 48864, USA (E-mail: banzhafw@msu.edu).

have to assume that the effective solutions include an effective primitive multiple times, which might not be the case in some applications (e.g., in program synthesis, a program repeats a primitive by looping [18]). It is tedious for human experts to design better FLs.

Regarding the performance gain brought by better fitness landscapes, one question naturally comes to mind: could we find better fitness landscapes for GP automatically, rather than manually designing them?

To answer this research question, this paper proposes a fitness landscape optimization (FLO) method that aggregates good GP solutions on the landscape by changing symbol indices. The paper takes linear genetic programming (LGP) as an example to verify the effectiveness of the proposed method since the linear representation of LGP is straightforward for demonstrating FLO. To the best of our knowledge, this paper is the first attempt to explicitly optimize FLs for symbolic search.

II. Background

A. Problem Definition of Stochastic Symbolic Search

A stochastic symbolic search problem consists of a search space $\mathbb{S}$ and a fitness function $\mathcal{F}$. The search space $\mathbb{S}$ is defined by $\mathbb{S} = \{\mathcal{S}, \mathcal{T}\}^m$ where $\mathcal{S}$ defines a set of available symbols, $\mathcal{T}$ defines a set of syntax rules, and m is the maximum size of a GP solution. The GP search problem is to find the optimal GP solution $g^* \in \mathbb{S}$ so that $\mathcal{F}$ is optimized. Supposing $\mathcal{F}$ is a minimizing problem, then we have

$$g^* = \arg\min_{g \in \mathbb{S}} \mathcal{F}(g).$$

GP searches possible solutions in $\mathbb{S}$ based on a certain order. The one-hop movement from one solution to another defines the neighborhood structure ν for a GP solution. ν is essentially a distance measure $\ominus$ between solutions (e.g., editing distance). In other words, solutions g_i and g_j are neighbors if $g_i \ominus g_j \leq \epsilon$, where ϵ is the threshold of a neighborhood. The fitness function $\mathcal{F}$, the search space $\mathbb{S}$, and the neighborhood structure ν constitute the fitness landscape $\mathcal{L}$ of a symbolic search problem $\mathcal{L} = \{\mathcal{F}, \mathbb{S}, \nu\}$.

B. Fitness Landscape in Genetic Programming

An FL is an important perspective for understanding the hardness of symbolic search problems. To investigate FLs, we have to define specific solution spaces and neighborhood structures. Given that this paper takes LGP as a case study, this subsection discusses existing studies of FLs in GP, which mainly focus on developing metrics to analyze the hardness of FLs and describe FL properties. In the experiments, we will apply the common metrics mentioned in this section to measure the hardness of optimized FLs.

Fitness distance correlation (FDC) is a representative of FL metrics. FDC measures the problem hardness by estimating the correlation between the distance and the fitness difference from global optima [19]. On an easy landscape, solutions are supposed to have better fitness when they are closer to known global optima, which means the fitness has a high correlation with the distance to the optimal solutions. Otherwise, the landscape is misleading. Normally, FDC divides the hardness of a search problem into three levels (suppose it is a minimizing problem) [20]:

1) An easy (or straightforward) fitness landscape: FDC>0.15, and FDC$=1$ is the ideal case.
2) A deceptive (or unknown) fitness landscape: $0.15 >$FDC> -0.15
3) A misleading fitness landscape: FDC< -0.15.

FDC has been shown to be a reliable FL metric of GP [21]–[25] and an effective way to analyze parameter configurations [26].

The *local optima network* is an FL analysis tool that enumerates all possible solutions in the search space to identify the local optima and their transitions [27], [28]. The local optima network visualizes an FL by a graph, in which vertices represent the local optima in the search space, and edges represent transitions (and their probability) between vertices. The characteristics of the graph such as the number of vertices and edges, and the cliquishness of a cluster (i.e., a connected sub-graph) show the characteristics of the FL (e.g., the connectivity of local optima and their distributions). For example, Đurasevic et al. [29] used the local optima network to analyze the effectiveness of different configurations of dimensionally-aware GP, and He and Neri [30] used the local optima to show the distribution of local optima.

However, it is infeasible to identify the global optima of problems with large search spaces in practice [31], [32]. To understand the FLs of GP benchmarks with large search spaces, existing studies developed several FL metrics based on the neighborhood of sampled solutions (i.e., neighborhood-based metrics).

Negative scope coefficient (NSC) is a neighborhood-based metric that measures the degree of "bad evolvability" (i.e., moving from good solutions to poor solutions) [33]. NSC first identifies the *fitness cloud* based on the fitness of sampled solutions and their neighbors [34]. The abscissas of the fitness cloud are the fitnesses of sampled solutions, and the ordinates of the fitness cloud are the fitnesses of their neighbors. Then, NSC partitions the fitness cloud into segments [35]. NSC gets the negative slopes among the mean values of segment abscissas and ordinates as its result. Normally, NSC is less than zero, and its absolute value indicates the degree of hardness of a problem.

Hu et al. [36]–[39] analyzed FLs of GP based on *robustness*, *evolvability*, and *accessibility* at three levels: genotypes, phenotypes, and fitness. Specifically, genotypic robustness indicates the fraction of neutral moves[1] caused by point mutations for a given genotype, genotypic (or a phenotype) evolvability indicates the proportion of non-neutral moves from a given genotype, and phenotypic accessibility indicates the propensity of mutating into a certain phenotype. Their results imply that robustness and evolvability are negatively correlated at the

[1]A neutral move is a movement between two solutions with the same fitness on FLs. A non-neutral move is a movement between two solutions with different fitnesses. Specifically, moving toward better fitness is known as "contributive move", and moving toward worse fitness is known as "destructive move".

genotypic level. Robust genotypes are normally hard to move into genotypes with another phenotype by a point mutation. Galván-López et al. [40] proposed to use *locality* [41] to measure the consistency between genotypic and phenotypic neighborhood structures. They assume that an FL with better consistency between genotypic and phenotypic neighborhood structures is easier for GP to search.

To sample solutions from FLs, several metrics perform a random walk on FLs. For example, Kinnear [4] used a *landscape autocorrelation* to measure the correlation of fitness over the random walk on an FL. Slaný and Sekanina [42] proposed two quantity metrics for *ruggedness* and *smoothness* based on the entropy of fitness over the random walk. However, these random walk-based metrics are not accurate enough to predict algorithm performance since in many problems they are highly dependent on the starting point of walking and it is difficult to perform *importance sampling* (more weight to sample good solutions) [6].

C. Improving Fitness Landscapes in Genetic Programming

Although very few studies explicitly optimize GP's FLs, many existing GP studies are essentially improving the FLs. To illustrate this idea, this subsection discusses example GP studies from the perspective of the three components of FLs.

1) Fitness Functions: The fitness function is a very problem-specific component in FLs. In symbolic regression problems, *root mean square error* is one of the common fitness functions to indicate GP's approximation performance. However, the root mean square error only considers pairwise errors in training data but does not consider the data distribution. To encourage GP to produce more concise and less overfit solutions, Haut et al. [43] proposed to use R^2, a measure of correlation coefficient, as the fitness function when training GP for symbolic regression tasks. Chen et al. [44] further verified the effectiveness of applying linear scaling with R^2 in GP for symbolic regression tasks.

Designing better fitness functions for a specific domain is non-trivial and tedious. To construct better fitness functions for a wider range of applications, multitask GP simultaneously optimizes several similar tasks, expecting that these similar tasks have synergistic fitness functions. Multitask GP has shown great potential in combinatorial optimization problems [15], [45] and classification problems [46], [47]. From a broader perspective, since fitness functions exert evolutionary pressure on GP by selection operators, advancing selection operators is an alternative way to improve fitness functions of GP [48], [49].

2) Solution Spaces: Each GP individual is a solution, and GP representations directly determine the solution space. There have been many GP representations [50]–[52]. For example, with the same primitive set, tree-based and linear representations have very different solution spaces because of the different topological structures of primitives [7], [53]. These representations have pros and cons for different problems [54]. Tree-based GP is good at parallelizing building block computation, while LGP is good at reusing building blocks. To reduce redundant solutions from solution spaces, existing studies also apply feature selection [16], [55] and grammar-guided techniques [56]–[58] to GP solution spaces.

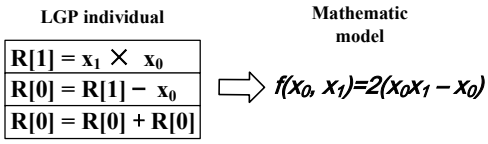


Fig. 1. An example LGP individual with three instructions.

3) Neighborhood Structures: Neighborhood structures define the neighborhood relationship among solutions. The neighborhood structures of existing GP methods are essentially genetic operators. Two neighboring GP solutions can reach each other by performing the genetic operator once. There are a huge number of existing studies proposing various genetic operators for GP based on different levels of information, such as the genotype-based [17], phenotype-based [59]–[61], and semantics-based [62], [63].

However, the performance of these genetic operators is limited by the "many-to-one" mapping (e.g., multiple genotypes map to one phenotype, and multiple phenotypes map to one semantics). The "many-to-one" mapping prevents GP from straightforwardly searching the solution space. In other words, it is uneasy for GP to identify a corresponding symbolic solution even when it has higher-level information (e.g., phenotypes and semantics). Moreover, designing genetic operators based on these levels of information needs a lot of domain knowledge and manual tuning. It is hard and expensive to refine FLs and extend these improvements to other domains.

D. Linear Genetic Programming

This paper takes LGP [53] as an example GP method to demonstrate the proposed FLO method. LGP represents symbolic solutions in the form of assembly programs. Each LGP individual is a sequence of register-based instructions. Fig. 1 shows an example of LGP individuals with three instructions, representing a mathematical formula $2(x_0x_1 - x_0)$. LGP first initializes registers by certain values (e.g., 1 in symbolic regression and low-level heuristic values in DJSS [54]). LGP sequentially executes the instructions to represent a computer program. The first register "R[0]" outputs the program result by default. Each instruction consists of four components, a destination register, a function, and two source registers. The function accepts the values in source registers as inputs and outputs the result to the destination register. LGP applies micro and macro mutation, and linear crossover to produce offspring. Specifically, micro mutation produces offspring by changing primitives in existing instructions, macro mutation by inserting or removing instructions, and linear crossover by swapping instruction segments over two LGP individuals.

Because of the use of registers, LGP naturally reuses intermediate results, which was shown to be effective in practice [54], [64]. In this paper, the linear representation of LGP individuals greatly facilitates our indexing of symbols and solutions by treating instructions as symbols and an LGP program as a symbolic solution. Theoretically, FLO is applicable to any GP representation that can be represented as

a list of symbols, among which the LGP representation is the most straightforward choice for this study.

III. Fitness Landscape Optimization

A. Main Idea

The main idea of fitness landscape optimization is to first index symbols into integers, and second to optimize the fitness landscape based on such symbol indices. We optimize the fitness landscape by rearranging the neighbors of symbolic solutions, i.e., by changing the indices of symbols. We do not optimize the fitness function or the solution space here since they are usually given by specific problems. Our method is generic for stochastic symbolic search methods whose solutions are represented as a list of symbols (e.g., assembly programs). Each symbol has a unique index, and each symbolic model is a vector of indices. Unlike existing GP studies that define neighborhoods based on genetic operators (e.g., crossover and mutation), we define the neighborhood structures of symbolic solutions based on the Euclidean distance of the index vectors.

We ordinally denote the indices of symbols as $\mathbb{I} = [\mathbb{I}_1, \mathbb{I}_2, ..., \mathbb{I}_l, ..., \mathbb{I}_n]^T$ where n is the number of symbols. We define an optimization objective function $F(\mathbb{I})$ for the fitness landscape. FLO is essentially an optimization problem

$$\mathbb{I}^* = \arg\min_{\mathbb{I}} F(\mathbb{I}).$$

The main goal of the objective $F(\mathbb{I})$ is to minimize the distance between good solutions, maximize the distance between good and poor solutions, and encourage the optimized indices to be consistent with domain knowledge. By this means, the number of local optima between global optima would be reduced, and the FL becomes less rugged. The symbol indices $\mathbb{I}$ represent a solution G_i by a mapping matrix θ_i, $G_i = \theta_i \mathbb{I}$, where

$$G_i = [G_{i1}, G_{i2}, ..., G_{ik}, ..., G_{im}]^T.$$

G_i is an ordinal encoding for a symbolic solution.

$$\theta_i = \mathbb{R}^{m \times n} = \begin{bmatrix} \theta_{i1} \\ \theta_{i2} \\ \vdots \\ \theta_{im} \end{bmatrix} = \begin{bmatrix} \theta_{i,1,1} & \theta_{i,1,2} & \cdots & \theta_{i,1,n} \\ \theta_{i,2,1} & \ddots & & \vdots \\ \vdots & & \theta_{i,k,l} & \vdots \\ \theta_{i,m,1} & \cdots & \cdots & \theta_{i,m,n} \end{bmatrix},$$

where $m (m \geq 1)$ is the maximum length of G_i. Each row of θ_i has at most one "1", and the rest of its elements are "0" (i.e., θ_{ik} is one-hot encoding). Since symbolic solutions might not have the maximum length m, G_i uses a placeholder "-1" to index those empty symbols.

To illustrate the idea, Fig. 2 shows a simple example of FLO. Suppose there are 12 possible symbols, denoted from **A** to **L**. These symbols are indexed by 0 to 11 initially (i.e., $\mathbb{I} = [0, 1, ..., 11]^T$). For the sake of simplicity, in this example, each program in the search space has one symbol (i.e., $m = 1$). We thus have 12 unique programs, denoted from G_1 to G_{12}. Specifically, $G_1 = [0]$, $G_2 = [1]$, $G_3 = [2]$, etc. We define the neighborhood structure as a Euclidean distance $||G_i - G_j||_2 \leq 1, i, j \in \{1, 2, ..., 12\}$. Fig. 2-(a) shows the initial FL of these programs. We can see that the initial FL is rugged. There are two optimal programs (supposing it is a minimizing problem) on the FL, "$G_1 = [0]$" and "$G_{12} = [11]$". To smoothen the FL and reduce the local optima, we aggregate good programs (e.g., G_1 and G_2) and separate good and poor programs (e.g., G_1 and G_6). For example, we let **A** and **L** have indices of 3 and 4 respectively. Consequently, G_1 has an index vector of "[3]", and G_2 has an index vector of "[4]", Fig. 2-(b) shows the optimized FL, where the symbol index $\mathbb{I} = [3, 2, 0, 1, ..., 9, 4]^T$, each symbol with a different index from the initial $\mathbb{I}$. The new $\mathbb{I}$ defines new neighbors for the programs. We can see that by aggregating good solutions and separating good and poor solutions, the FL becomes smoother and more "cone-like" (i.e., the distance to the optimal solutions is highly correlated to the fitness discrepancy) than the initial one. The number of local optima is also reduced. It is easier to search for optimal solutions on the optimized FL than on the initial one.

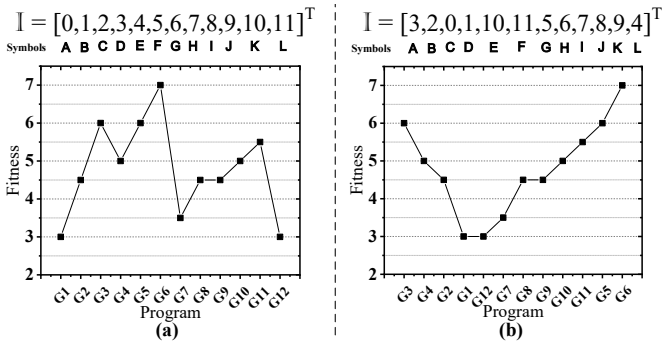


Fig. 2. A simple example of FLO. (a) the initial FL is rugged; (b) the optimized FL is cone-like. By optimizing the indices of symbols that lead to different neighbors for each solution, we make the FL easier to search.

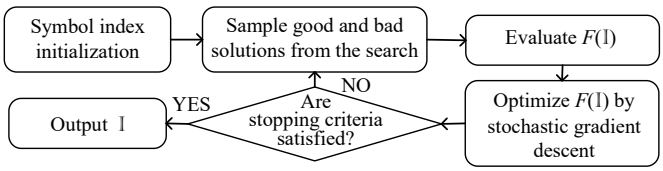


Fig. 3. The overall framework of FLO.

B. Overall Framework

The overall framework of FLO is shown in Fig. 3[2]. We first enumerate all the possible symbols and create an initial symbol index $\mathbb{I}$. FLO samples good and bad solutions from the search and then evaluates $F(\mathbb{I})$ based on the sampled solutions. FLO applies a stochastic gradient descent method to optimize the indices based on the sampled solutions. We denote the set of sampled good and bad solutions as $\mathbb{B}$ and $\mathbb{B}_{lose}$, respectively. FLO repeats the optimization until it reaches the stopping criteria.

C. Optimization Objectives

This section formulates our optimization objectives $F(\mathbb{I})$. Based on the main idea of optimization objectives, there are

[2]Our code will be public after paper acceptance.

three sub-objectives: inner distance between good solutions, inter distance between good and poor solutions, and the consistency with domain knowledge. They are denoted as $D(\mathbb{I})$, $D_{lose}(\mathbb{I})$, and $E(N(\mathbb{I}))$, respectively. $F(\mathbb{I})$ combines these three objectives linearly, as shown in Eq. (1) where $\alpha_1 = \alpha_2 = \alpha_3$ and $\sum_{i=1}^{i=3} \alpha_i = 1$ by default.

$$F(\mathbb{I}) = \frac{\alpha_1}{D(\mathbb{I}_0)} D(\mathbb{I}) + \frac{\alpha_2 \times D_{lose}(\mathbb{I}_0)}{D_{lose}(\mathbb{I})} + \frac{\alpha_3}{E(N(\mathbb{I}_0))} E(N(\mathbb{I})), \quad (1)$$

subject to

$$\mathbb{I}_l \in N; \mathbb{I}_l < n; \mathbb{I}_i \neq \mathbb{I}_j \text{ if } i \neq j.$$

The three constraints ensure that the range and the uniqueness of the indices. Because these sub-objectives have different data scales, they are normalized to $[0, 1]$ by $D(\mathbb{I}_0)$, $D_{lose}(\mathbb{I}_0)$, and $E(N(\mathbb{I}_0))$, where $\mathbb{I}_0$ are the indices before each optimization.

1) Inner Distance between Good Solutions:

$$D(\mathbb{I}) = \frac{1}{|\mathbb{B}|^2} \sum_{a=1}^{|\mathbb{B}|} \sum_{b=1}^{|\mathbb{B}|} ||G_a - G_b||_2^2. \quad (2)$$

$D(\mathbb{I})$ formulates the Euclidean distance between good solutions, as shown in Eq. (2). To simplify the derivation, we omit the square root in the Euclidean distance (i.e., squaring the L^2 norm). Objectives $D_{lose}(\mathbb{I})$ and $E(N(\mathbb{I}))$ also apply the squared L^2 norm for simplifying the derivation. For all pairs of good solutions in $\mathbb{B}$, $D(\mathbb{I})$ sums the squared Euclidean distance and calculates their average.

To clarify the relationship between $\mathbb{I}$ and $D(\mathbb{I})$, we denote $||G_a - G_b||_2^2$ as $Q_{ab}(\mathbb{I})$, and thus $D(\mathbb{I}) = \frac{1}{|\mathbb{B}|^2} \sum_{a=1}^{|\mathbb{B}|} \sum_{b=1}^{|\mathbb{B}|} Q_{ab}(\mathbb{I})$. By substituting $G_{ik} = \theta_{ik}\mathbb{I}$, we have:

$$\begin{aligned} Q_{ab}(\mathbb{I}) &= ||G_a - G_b||_2^2 = \sum_{k=1}^{m} (G_{ak} - G_{bk})^2 \\ &= \sum_{k=1}^{m} (\theta_{a,k}\mathbb{I} - \theta_{b,k}\mathbb{I})^2. \end{aligned} \quad (3)$$

Because θ_{ik} is one-hot mapping, $Q_{ab}(\mathbb{I})$ is further simplified.

$$Q_{ab}(\mathbb{I}) = \sum_{k=1}^{m} (\sum_{l=1}^{n} (\theta_{a,k,l} - \theta_{b,k,l})\mathbb{I})^2.$$

Note that since G_{ak} and G_{bk} may be "-1", which indicates an empty corresponding symbol, the subtraction of $G_{ak} - G_{bk}$ is redefined as:

$$G_{ak} - G_{bk} = \begin{cases} G_{ak} - G_{bk} & G_{ak} \geq 0 \text{ and } G_{bk} \geq 0 \\ 0 & G_{ak} < 0 \text{ and } G_{bk} < 0 \\ 1 & \text{otherwise} \end{cases}.$$

2) Inter Distance between Good and Bad Solutions:

$$\begin{aligned} D_{lose}(\mathbb{I}) &= \frac{1}{|\mathbb{B}||\mathbb{B}_{lose}|} \sum_{a \in \mathbb{B}}^{|\mathbb{B}|} \sum_{b \in \mathbb{B}_{lose}}^{|\mathbb{B}_{lose}|} ||G_a - G_b||_2^2 \\ &= \frac{1}{|\mathbb{B}||\mathbb{B}_{lose}|} \sum_{a \in \mathbb{B}}^{|\mathbb{B}|} \sum_{b \in \mathbb{B}_{lose}}^{|\mathbb{B}_{lose}|} Q_{ab}(\mathbb{I}). \end{aligned} \quad (4)$$

In contrast to $D(\mathbb{I})$, $D_{lose}(\mathbb{I})$ formulates the Euclidean distance between good and poor solutions, as shown in Eq. (4). $D_{lose}(\mathbb{I})$ considers poor solutions in the calculation (i.e., $\mathbb{B}_{lose}$). To make $D_{lose}(\mathbb{I})$ more comprehensive, there is an archive that records the poor solutions over the search. $\mathbb{B}_{lose}$ samples poor solutions from both the current search stage and the historical archive. $\mathbb{B}_{lose}$ have the same size with $\mathbb{B}$.

3) Domain Knowledge Consistency:

$$E(N(\mathbb{I})) = ||N_0 \otimes N_0 - \frac{1}{n^2} N(\mathbb{I}) \otimes N(\mathbb{I})||_2^2. \quad (5)$$

$E(N(\mathbb{I}))$ is defined as the difference between the domain knowledge and the existing indices, as shown in Eq. (5). Smaller differences imply that the symbol indices are more consistent with the domain knowledge. Eq. (5) squares the L^2 norm to simplify the derivation, and applies the element-wise production (denoted as $\otimes$). N_0 is a predefined distance matrix whose elements are the distance between symbols based on the domain knowledge. For example, character $\mathcal{A}$ is closer to $\mathcal{B}$ than to $\mathcal{Z}$ based on the alphabetical order. We denote N_0 as

$$\begin{aligned} N_0 = \mathbb{R}^{n \times n} &= [d_{N,1}, \cdots, d_{N,l}, \cdots, d_{N,n}] \\ &= \begin{bmatrix} d_{N,1,1} & \cdots & d_{N,l,1} & \cdots & d_{N,n,1} \\ d_{N,1,2} & \ddots & & & \vdots \\ \vdots & & & \ddots & \vdots \\ d_{N,1,n} & & \cdots & & d_{N,n,n} \end{bmatrix}. \end{aligned}$$

In this paper, we define $d_{N,a,b}$ as the product of the editing distance ($D_{g,a,b}$) and the semantic distance ($D_{s,a,b}$) of instructions a and b. $d_{N,a,b} = \frac{D_{g,a,b}}{D_{g,max}} \cdot \frac{D_{s,a,b}}{D_{s,max}}$ where $D_{g,max}$ and $D_{s,max}$ are the maximum D_g and D_s among the possible instructions. The semantic distance is defined as the Euclidean distance between instruction outputs. The instructions get their outputs based on ten randomly sampled input instances. The number of sampled input instances is set based on our prior investigation, which did not exert a significant impact on final performance. As these instruction outputs can be reused for different problems, they are not counted in the total fitness evaluation.

$N(\mathbb{I})$ is a distance matrix between the current indices.

$$N(\mathbb{I}) = \mathbb{R}^{n \times n} = [\Delta_1\mathbb{I}, \cdots, \Delta_l\mathbb{I}, \cdots, \Delta_n\mathbb{I}],$$

where

$$\Delta_l = \mathbb{R}^{n \times n} = \begin{bmatrix} & & & l^{th}column & \\ -1 & & & 1 & \\ & -1 & & 1 & \\ & & \ddots & 1 & \\ & & & 0 & \\ & & & 1 & -1 \end{bmatrix}.$$

Because $d_{N,i,j} \in [0, 1]$, we normalize $N(\mathbb{I}) \otimes N(\mathbb{I})$ by $\frac{1}{n^2}$. Based on N_0 and $N(\mathbb{I})$, Eq. (5) can be extended as $E(N(\mathbb{I})) = \sum_j^n \sum_i^n \left(d_{N,i,j}^2 - \frac{1}{n^2}(\mathbb{I}_j - \mathbb{I}_i)^2\right)^2$.

D. Stochastic Gradient Descent

We optimize $F(\mathbb{I})$ by a stochastic gradient descent method. Specifically, the new indices

$$\mathbb{I}' = \lfloor \mathbb{I} - sign(\frac{\partial F}{\partial \mathbb{I}}) - U(\sigma n) \times \frac{\partial F}{\partial \mathbb{I}} \rfloor, \quad (6)$$

where σ is the maximum step size and $U(\sigma n)$ returns a uniformly random float value in $[0, \sigma n]$. σ has a range of $(0, 1]$ and multiplies by n to represent actual step size. We apply stochastic step size here to let our search be capable of jumping out of local optima and fine-tune indices by probability. $sign(\cdot)$ returns 1 for positive inputs, 0 for zero inputs, and -1 otherwise. $-sign(\frac{\partial F}{\partial \mathbb{I}})$ ensures that a symbol index at least moves one unit along the negative gradient. FLO

performs a flooring operation on $\mathbb{I}$ to ensure integer indices. If $F(\mathbb{I}') < F(\mathbb{I})$, $\mathbb{I}$ is updated by $\mathbb{I}'$ (i.e., $\mathbb{I} = \mathbb{I}'$). The stochastic gradient descent iterates until it reaches the maximum iteration times. Based on Eq. (1), we have:

$$\frac{\partial F}{\partial \mathbb{I}} = \frac{\alpha_1}{D(\mathbb{I}_0)}\frac{\partial D}{\partial \mathbb{I}} + \frac{\alpha_2 \times D_{lose}(\mathbb{I}_0)}{-D_{lose}(\mathbb{I})^2}\frac{\partial D_{lose}}{\partial \mathbb{I}} + \frac{\alpha_3}{E(N(\mathbb{I}_0))}\frac{\partial E}{\partial \mathbb{I}}. \quad (7)$$

The partial derivation of $D(\mathbb{I})$ is shown as Eq. (8).

$$\frac{\partial D}{\partial \mathbb{I}} = \left[\frac{\partial D}{\partial \mathbb{I}_1}, \cdots, \frac{\partial D}{\partial \mathbb{I}_l}, \cdots, \frac{\partial D}{\partial \mathbb{I}_n}\right]^T, \quad (8)$$

where

$$\begin{aligned}\frac{\partial D}{\partial \mathbb{I}_l} &= \frac{1}{|\mathbb{B}|^2}\sum_{a=1}^{|\mathbb{B}|}\sum_{b=1}^{|\mathbb{B}|}\frac{\partial Q_{ab}}{\partial \mathbb{I}_l} \\ &= \frac{1}{|\mathbb{B}|^2}\sum_{a=1}^{|\mathbb{B}|}\sum_{b=1}^{|\mathbb{B}|}(\sum_{k}^{m} 2(G_{ak} - G_{bk})(\theta_{a,k,l} - \theta_{b,k,l})). \quad (9)\end{aligned}$$

Based on Eq. (9), we can know that when a and b are duplicated solutions, the gradient of $\frac{\partial D}{\partial \mathbb{I}_l}$ will vanish. To avoid the gradient vanishing, we should eliminate duplicated solutions in $\mathbb{B}$ and sample diverse good solutions.

Similarly, the partial derivation of $D_{lose}(\mathbb{I})$ is shown as Eq. (10).

$$\frac{\partial D_{lose}}{\partial \mathbb{I}} = \left[\frac{\partial D_{lose}}{\partial \mathbb{I}_1}, \cdots, \frac{\partial D_{lose}}{\partial \mathbb{I}_l}, \cdots, \frac{\partial D_{lose}}{\partial \mathbb{I}_n}\right]^T, \quad (10)$$

where

$$\begin{aligned}\frac{\partial D_{lose}}{\partial \mathbb{I}_l} &= \frac{1}{|\mathbb{B}|}\frac{1}{|\mathbb{B}_{lose}|}\sum_{a\in\mathbb{B}}^{|\mathbb{B}|}\sum_{b\in\mathbb{B}_{lose}}^{|\mathbb{B}_{lose}|}\frac{\partial Q_{ab}}{\partial \mathbb{I}_l} \\ &= \frac{1}{|\mathbb{B}|}\frac{1}{|\mathbb{B}_{lose}|}\sum_{a\in\mathbb{B}}^{|\mathbb{B}|}\sum_{b\in\mathbb{B}_{lose}}^{|\mathbb{B}_{lose}|} \\ &\quad \left[\sum_{k}^{m} 2(G_{ak} - G_{bk})(\theta_{a,k,l} - \theta_{b,k,l})\right]. \quad (11)\end{aligned}$$

As $\mathbb{B}$ and $\mathbb{B}_{lose}$ are unlikely overlapped (if they are overlapped, just simply remove the overlapped part), Eq. (11) is non-zero.

The partial derivation of $E(N(\mathbb{I}))$ is shown as Eq. (12).

$$\frac{\partial E}{\partial \mathbb{I}} = \left[\frac{\partial E}{\partial \mathbb{I}_1}, \cdots, \frac{\partial E}{\partial \mathbb{I}_l}, \cdots, \frac{\partial E}{\partial \mathbb{I}_n}\right]^T, \quad (12)$$

where

$$\begin{aligned}\frac{\partial E}{\partial \mathbb{I}_l} &= \sum_{j}^{n} 2\left[d_{N,l,j}^2 - \frac{1}{n^2}(\mathbb{I}_j - \mathbb{I}_l)^2\right]\frac{2}{n^2}(\mathbb{I}_j - \mathbb{I}_l) \\ &\quad + \sum_{i}^{n} 2\left[d_{N,i,l}^2 - \frac{1}{n^2}(\mathbb{I}_l - \mathbb{I}_i)^2\right]\frac{2}{n^2}(\mathbb{I}_i - \mathbb{I}_l). \quad (13)\end{aligned}$$

Note that in Eq. (13), the first item is not zero only when $i = l$ and $j \neq l$, and the second item is not zero when $j = l$ and $i \neq l$. We can simplify Eq. (13) by assuming $d_{N,i,l} = d_{N,l,i}$ and denoting j in the first item as i without loss of generality, as shown in Eq. (14)

$$\begin{aligned}\frac{\partial E}{\partial \mathbb{I}_l} &= \frac{4}{n^2}\left[\sum_{i}^{n}\left[d_{N,l,i}^2 - \frac{1}{n^2}(\mathbb{I}_i - \mathbb{I}_l)^2\right](\mathbb{I}_i - \mathbb{I}_l)\right. \\ &\quad \left. + \sum_{i}^{n}\left[d_{N,i,l}^2 - \frac{1}{n^2}(\mathbb{I}_l - \mathbb{I}_i)^2\right](\mathbb{I}_i - \mathbb{I}_l)\right] \\ &= \frac{8}{n^2}\left[\sum_{i}^{n}\left[d_{N,l,i}^2 - \frac{1}{n^2}(\mathbb{I}_i - \mathbb{I}_l)^2\right](\mathbb{I}_i - \mathbb{I}_l)\right]. \quad (14)\end{aligned}$$

Algorithm 1: `Fitness Landscape Optimization`

Input: A fitness landscape $\mathcal{L} = \{\mathcal{F}, \mathbb{S}, \nu\}$ where $\mathbb{S} = \{\mathcal{S}, \mathcal{T}\}^m$
Output: Optimized symbol indices $\mathbb{I}$

```
I ← initialize the indices of symbols in S.
while stopping criteria are not satisfied do
    Sample solutions from S and categorize them into good solutions B and bad solutions B_lose based on F.
    μ(I) ← get the used symbols in B.
    Evaluate F(I).
    /* stochastic gradient descent */
    for j ← 1 to 20 do
        Evaluate ∂F/∂I if I is changed.
        I' ← ⌊I − sign(∂F/∂I) − U(σn) × ∂F/∂I⌋
        Evaluate F(I').
        if F(I') < F(I) then
            /* I ← I' */
            I_l ← I'_l, l ∈ μ(I) in a random order.
            I_l ← I'_l, l ∉ μ(I) in a random order.
            F(I) ← F(I').
Return I;
```

In practice, $|\mathbb{I}_i - \mathbb{I}_j|$ is not important if $\mathbb{I}_i$ and $\mathbb{I}_j$ are not effective symbols. So, we further simplify Eq. (14) as:

$$\frac{\partial E}{\partial \mathbb{I}_l} = \frac{8}{n^2}\left[\sum_{i\in\mu(\mathbb{I})}^{|\mu(\mathbb{I})|}\left[d_{N,l,i}^2 - \frac{1}{n^2}(\mathbb{I}_i - \mathbb{I}_l)^2\right](\mathbb{I}_i - \mathbb{I}_l)\right], \quad (15)$$

where $\mu(\mathbb{I})$ are the used symbols in good solutions $\mathbb{B}$.

$\frac{\partial F}{\partial \mathbb{I}}$ is normally very sparse (i.e., there are a large number of zero or nearly zero elements) when n is large after normalization, which greatly limits the search efficiency of the stochastic gradient descent. To improve the search efficiency, we adjust $\frac{\partial F}{\partial \mathbb{I}}$ by

$$\frac{\partial F}{\partial \mathbb{I}_l} = sign(\frac{\partial F}{\partial \mathbb{I}_l})/(-\ln(\left|\frac{\partial F}{\partial \mathbb{I}_l}\right|)).$$

To satisfy the constraints in Eq. (1) and prioritize the effective symbols (indicated by $\mu(\mathbb{I})$), we first update $\mathbb{I}_l(l \in \mu(\mathbb{I}))$ one-by-one in a random order, and then update $\mathbb{I}_l(l \notin \mu(\mathbb{I}))$ one-by-one in a random order based on Eq. (6). If $\mathbb{I}'_i = \mathbb{I}'_j(i \neq j)$, we assign $\mathbb{I}'_j$ a random integer ranging between 0 and $n-1$ until $\mathbb{I}'_j$ is unique in $\mathbb{I}'$.

Alg. 1 shows the pseudo-code of FLO. Given a fitness landscape $\mathcal{L}$, we first initialize the indices of available symbols in $\mathcal{S}$. We then perform solution sampling and stochastic gradient descent to optimize $\mathcal{L}$ until the stopping criteria are satisfied. Specifically, we categorize the sampled solutions into good solutions $\mathbb{B}$ and bad solutions $\mathbb{B}_{lose}$, respectively, by the fitness function $\mathcal{F}$. We also collect the used symbols $\mu(\mathbb{I})$ in good solutions. After that, we optimize $\mathcal{L}$ by updating the symbol indices $\mathbb{I}$ which essentially changes the neighborhood structure ν. We stop the stochastic gradient descent after 20 iterations, which has been shown to be long enough in our prior experiments. If the new objective function $F(\mathbb{I}')$ is smaller than the original objective value $F(\mathbb{I})$, we use $\mathbb{I}'$ to update $\mathbb{I}$. The final symbol indices are output as the optimized indices.

TABLE I
TEST PROBLEMS

Problem	Mathematic model	Primitives	#instructions m	Number of possible solutions n_g
Finger	$f(x_0) = x_0 \times x_0 - x_0$	$\{+, -, \times, R_0, x_0\}$	2	$(1 \times 3 \times 2 \times 2)^2 = 144$
Toy	$f(x_0, x_1, x_2, x_3) = x_0 \times (x_1 + x_2)$	$\{+, -, \times, \div, R_0, R_1, R_2, x_0, x_1, x_2, x_3\}$	2	$(3 \times 4 \times 7 \times 7)^2 = 345744$
R1	$f(x_0) = \frac{(x_0+1)^3}{x_0^2 - x_0 + 1}$	$\{+, -, \times, \div, \sqrt{\lvert\cdot\rvert}, \ln(\lvert\cdot\rvert), R_0, x_0\}$	4	$(1 \times 6 \times 2 \times 2)^4 = 331776$
$\langle$Tmean, 0.85$\rangle$	unknown	$\{+, -, \max, R_0, R_1, PT, OWT\}$	3	$(2 \times 3 \times 4 \times 4)^3 = 373248$

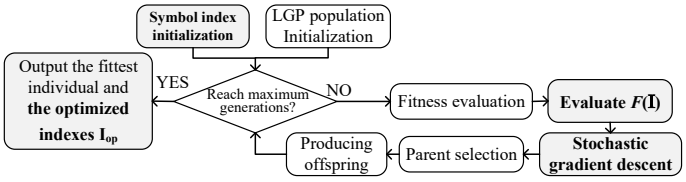


Fig. 4. The evolutionary framework of LGP with FLO. The components of FLO are highlighted in grey and bold.

E. Applying FLO to LGP

This paper applies FLO to LGP to verify the effectiveness of the proposed method. LGP here serves as the sampling method in FLO to obtain good and bad solutions from the search space. Fig. 4 shows the evolutionary framework of basic LGP with FLO. LGP searches for symbolic solutions by iteratively producing offspring based on good solutions. After fitness evaluation, we sort LGP individuals based on fitness and obtain up-to-date good and bad solutions (i.e., $\mathbb{B}$ and $\mathbb{B}_{lose}$) to perform FLO. The stopping criterion of LGP is the maximum number of generations. The fittest LGP individual and the optimized symbol indices are outputted at the end of LGP evolution.

IV. EXPERIMENT DESIGN

Our experiments verify the effectiveness of the proposed FLO method by 1) investigating the hardness reduction of the optimized landscapes of LGP and 2) analyzing the patterns of the optimized FLs within the context of LGP. Specifically, we measure the hardness reduction by four common FL metrics and analyze the patterns by visualizing the optimized FLs.

A. Performance Measure

To comprehensively analyze the hardness of landscapes, we apply four common FL analysis metrics in existing GP studies together, including fitness distance correlation (FDC) [19], negative scope coefficient (NSC) [33], robustness (RBS) [37], and evolvability (EVO) [6]. Specifically, FDC measures the degree that an FL looks like a "cone" (i.e., a high correlation between the distance to optima and the fitness difference from optima). NSC measures the degree of "bad evolvability". RBS measures the probability of a neutral move. EVO measures the probability of moving toward a better neighbor. Larger values of the four metrics mean an easier FL, in other words, an easier search problem.

B. Test Problems

We verify the effectiveness of the proposed method in supervised learning problems and the learn-to-optimize combinatorial optimization problems. Specifically, we select three symbolic regression problems, i.e., supervised learning problems, and a ubiquitous dynamic combinatorial optimization problem, i.e., dynamic job shop scheduling (DJSS), as our tested problems since GP has been successfully applied to these two tasks [13], [14]. The comparison between the proposed method and advanced methods in these two tasks can imply the potential generality of the proposed method. The details of the problems are shown in Table I. LGP searches for mathematical models for the symbolic regression problems based on the labeled data, and learns decision policies (i.e., dispatching rules) to optimize the DJSS problem based on the simulation performance. Finger and Toy are two naive symbolic regression problems with different difficulty levels. R1 and $\langle Tmean, 0.85\rangle$ are common benchmark problems in symbolic regression and DJSS [15], [63]. $\langle Tmean, 0.85\rangle$ indicates a DJSS scenario optimizing the mean tardiness of job shops with a utilization level of 0.85. The utilization level indicates the rate of new job arrival, and the value of 0.85 indicates a busy job shop. The DJSS scenario is essentially a distribution that samples the simulation instances for training and testing. LGP searches a decision policy to make instant decisions for the dynamic events in DJSS [54]. The "Mathematic model" column indicates the ground truth symbolic solutions of the investigated problems. The "Primitives" column lists the functions and terminals for LGP, where R_i denotes the registers in LGP instructions and x_i denotes the input dimension. "PT" and "OWT" are the input features in DJSS, which indicate the processing time of operations and their waiting time.

We strictly limit the search space of the four tested problems so that the search space can be enumerated in an acceptable time for the visualization of FLs. Both Finger and Toy are symbolic regression problems with a maximum program size of 2 instructions. In the Toy problem, the input data has a redundant dimension (i.e., x_3) to increase problem hardness. To limit the number of possible solutions in R1 and $\langle Tmean, 0.85\rangle$, we only retain necessary primitives and constrain the maximum program size of 4 and 3 instructions, respectively. Table I shows the approximated number of all the possible solutions of these four tested problems. We count the number of possible solutions n_g based on the following formula.

$$n_g = (r_d \times n_f \times (r_s + n_x)^2)^m,$$

where r_d is the number of destination registers, n_f is the number of functions, r_s is the number of source registers,

n_x is the number of input features, and m is the maximum program size. The fitness function of the symbolic regression problems is root-mean-square-error (RMSE), and the fitness function of $\langle Tmean, 0.85\rangle$ is the mean tardiness over simulation. Although these example test problems have very limited search spaces, they cover different kinds of domains, and different numbers of instructions to make the experiments more comprehensive. Note that our proposed method can scale to symbolic models with more than 4 instructions. For example, the symbolic models in Section VI have 50 to 100 instructions.

C. Compared Methods

The compared methods in our experiments are essentially the neighborhood structures of FLs. Given the same fitness functions and solution spaces, different neighborhood structures likely will lead to different hardness of FLs. The neighborhood structures in this paper include 1) the Euclidean distance based on optimized symbol indices and 2) the genetic operators of LGP. We verify the effectiveness of FLO by comparing the hardness of FLs with the optimized symbol indices with the hardness of FLs whose neighborhood structures are defined by genetic operators of LGP.

We compare FLO with two LGP genetic operators, free macro mutation (denoted as "freemut") [53] and frequency-based macro mutation (denoted as "freqmut"). These two genetic operators define the neighborhood structures of compared FLs. Freemut is a basic mutation operator of LGP, serving as a baseline. It defines the neighbors of an LGP individual by inserting or removing a random instruction into/from the individual. Freqmut is a self-adaptive genetic operator that adapts over the generations. It defines the neighbors of an LGP individual based on the primitive frequency in the top-K individuals (K=10% in our experiments). The assumption that primitive frequency implies importance of primitives is common in advanced GP methods [17], [65]. Freqmut serves as an advanced manually designed neighborhood structure. These two genetic operators are used for identifying neighbors of LGP individuals when we measure the FL analysis metrics.

The neighborhood structure of FLO is defined based on the Euclidean distance of index vectors. Two LGP individuals G_a and G_b are neighbors if $||G_a - G_b||_2 \leq \epsilon$. Since freeemut and freqmut only vary one instruction when identifying neighbors, we also only vary one instruction when identifying neighbors in FLO.

In an independent run, the FL metrics analyze the hardness of FLs of the three compared neighborhood structures at the final generation of LGP search. The three methods compared share the same LGP search as the sampling methods, which ensures the same sampled solutions in analyzing FL hardness. To compare the performance comprehensively, each method is tested in 50 independent runs on each test problem.

D. Parameter Settings

FLO has two main parameters, the number of sampled good solutions from the population B and the maximum step size in the stochastic gradient descent σ. We set $B = 10$ and set $\sigma = 0.1$. Specifically, to sample diverse good solutions in $\mathbb{B}$, we consider the top-B fitness values in the population, each getting one individual, and vice versa for $\mathbb{B}_{lose}$. The threshold of the neighborhood in FLO ϵ is not a parameter in FLO. It is a problem-specific parameter when calculating FL metrics. ϵ is set to approximately 5% of the total number of instructions, except 5 for Finger which only has 12 possible instructions. The parameters of the basic LGP evolution keep the same as those of basic LGP for solving symbolic regression and dynamic job shop scheduling [53], [54]. Each FLO iterates up to 20 times and is performed every two LGP generations.

TABLE II
FRIEDMAN TEST OVER THE COMPARED METHODS ON THE FL METRICS OF THE FOUR EXAMINED PROBLEMS

Metrics		freemut	freqmut	FLO
FDC	mean rank	2.375	2.625	**1**
	p-value		0.038	
NSC	mean rank	2.25	2	**1.75**
	p-value		0.779	
RBS	mean rank	2.25	**1.5**	2.25
	p-value		0.472	
EVO	mean rank	1.75	2.75	**1.25**
	p-value		0.174	

TABLE III
THE MEAN METRIC VALUES (AND STANDARD DEVIATION) ON THE TESTED PROBLEMS

Problems	Metrics	freemut	freqmut	FLO (Ours)
Finger	FDC	0.205 (0) −	0.205 (0) −	**0.361 (0.126)**
	NSC	−3.098 (0.001) −	**−1.934 (0.238)** ≈	−1.951 (0.364)
	RBS	0.381 (0) −	0.339 (0.014) −	**0.415 (0.01)**
	EVO	**0.86 (0)** +	0.826 (0.014) ≈	0.811 (0.032)
Toy	FDC	0.02 (0.106) −	0.056 (0.155) −	**0.279 (0.206)**
	NSC	−4.087 (13.19) ≈	−4.082 (10.19) ≈	**−1.914 (1.254)**
	RBS	0.263 (0.072) ≈	**0.332 (0.055)** +	0.257 (0.05)
	EVO	0.587 (0.049) −	0.506 (0.065) −	**0.615 (0.062)**
R1	FDC	0.066 (0.086) −	−0.029 (0.061) −	**0.131 (0.104)**
	NSC	**−12.42 (11.43)** ≈	−27.9 (30.69) ≈	−14.95 (19.02)
	RBS	0.326 (0.087) −	**0.615 (0.135)** +	0.477 (0.024)
	EVO	0.584 (0.079) −	0.285 (0.135) −	**0.81 (0.034)**
⟨Tmean,0.85⟩	FDC	0.072 (0.086) −	0.046 (0.082) −	**0.184 (0.128)**
	NSC	−21.13 (70.12) ≈	−11.83 (42.6) ≈	**−11.23 (37.59)**
	RBS	0.699 (0.037) ≈	**0.765 (0.034)** +	0.695 (0.019)
	EVO	0.182 (0.043) −	0.092 (0.03) −	**0.407 (0.078)**

V. EXPERIMENT RESULTS

A. FL Hardness

This section analyzes the hardness of FLs defined by the compared neighborhood structures at the final LGP generation. Table II shows the overall performance of the compared methods on the four FL metrics based on the Friedman test with a significance level of 0.05. In Table II, a small rank value indicates a large metric value (i.e., an easy FL). We can see that FLO has a significantly easier FL in terms of FDC over the tested problems. This implies that FLO successfully optimizes the FLs to be more cone-like than the compared methods. Table II also shows that the FLs of FLO have a better mean NSC and EVO than freemut and freqmut, which implies a higher probability of moving toward better neighbors on the optimized FLs.

Table III shows the mean metric values (and their standard deviations) of the compared methods for each problem. We apply the Wilcoxon rank-sum test with a significance level

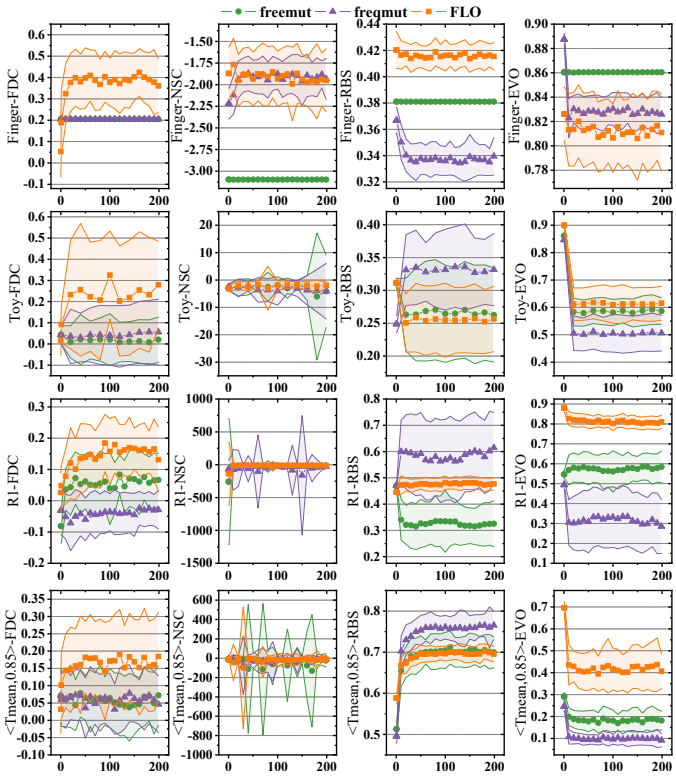


Fig. 5. FL metrics over generations. X-axis: the number of LGP generations, Y-axis: the metric values of a problem. The four columns from the left to right are FDC, NSC, RBS, and EVO.

of 0.05 to analyze these metric values. "+" indicates a significantly better metric value, "−" indicates a significantly worse metric value, and "≈" indicates a statistically similar metric value with the proposed method (i.e., FLO here). The best mean metric values are highlighted in bold. Aligned with Table II, the FLs of FLO have significantly better FDC values on the four tested problems. In terms of NSC, the optimized FLs of FLO show competitive hardness with the compared methods. In terms of RBS and EVO, although the FLs of freqmut have better RBS values (i.e., a high probability of neutral move) in Toy, R1, and ⟨Tmean,0.85⟩, they sacrifice the probability of moving to better neighbors (i.e, significantly worse EVO values than FLO in the last three tested problems). To conclude, the optimized FLs of FLO are significantly easier than the ones of baseline and advanced neighborhood structures in terms of FDC and at least competitive with the compared methods in terms of the other three metrics. The results show a very encouraging optimization performance of FLO.

B. Metrics Over Generations

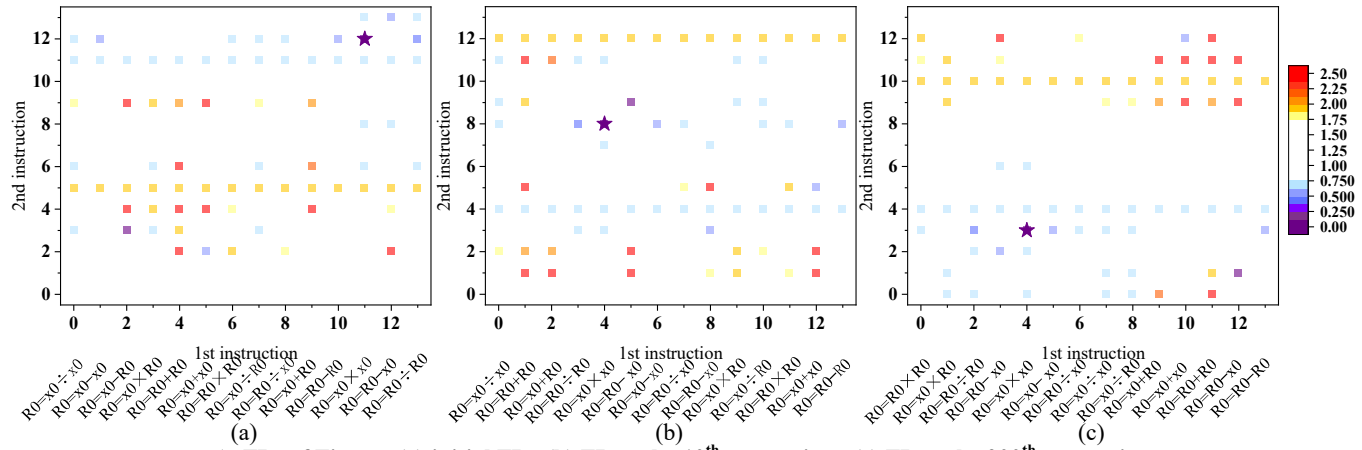
To understand the optimization process of FLO, this section investigates the change of FL hardness over LGP search, as shown in Fig. 5. Each column in Fig. 5 shows a certain metric on the four tested problems. In terms of FDC (i.e., the first column in Fig. 5), FLO substantially improves FDC values over generations and maintains at a significantly higher level than freemut and freqmut. In terms of NSC, the three compared methods all stay at a similar level over the whole evolution. But we can see that FLO averagely has a smaller standard deviation than the compared methods in the last three problems (i.e., Toy, R1, and ⟨Tmean,0.85⟩), which implies a more stable FL hardness of FLO. The curves of RBS seem problem-specific since the three compared methods show very different behaviors on the tested problems. But benefit from the frequency-based mutation that prefers sampling primitives in best-to-the-run solutions, freqmut averagely improves the probability of a neutral move in most of the tested problems. In terms of EVO, although most of the curves decline over the generations since there are fewer and fewer better neighbors when the LGP population moves to optimal solutions, the FLs of FLO often maintain a higher level of EVO than the compared methods.

C. Pattern Analyses on FLs

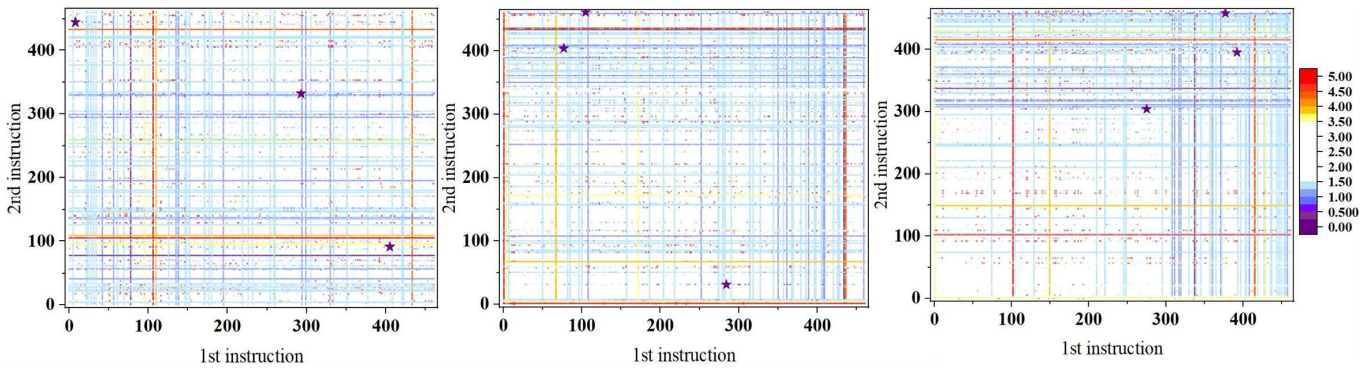
To intuitively investigate the FLs obtained by FLO, we visualize the example FLs and discuss their patterns in this section. Specifically, we show the example FLs obtained at different stages of LGP search on Finger, Toy, and ⟨Tmean,0.85⟩ problems. The solutions in these three problems have up to three instructions, each instruction for one dimension, which is straightforward for visualization. Fig. 6 shows the scatter plots of the example FLs. Each point on the FLs is a symbolic solution. The color of the points represents the fitness of the solution. To highlight the patterns of the solutions, we color the solutions with the best 25% fitness and with the worst 25% fitness, by cool and warm tones respectively. We leave the remaining solutions with moderate fitness as transparent to keep the visualization results sharp and clean. We also highlight the optimal solutions by purple stars. The optimal solutions are identified after enumerating all possible solutions. Fig. 6 has three sub-figures, A, B, and C, each for a selected problem. The instructions at the bottom of sub-figure A are the corresponding instructions for each index, where "R0" stands for the first (and the only one) register, and "x0" stands for the first input feature. The sub-figures in each problem visualize the dynamics of FLO from initial FLs, to the FLs at the 10^{th} generations, and to the final FLs. To show the FL of ⟨Tmean, 0.85⟩ at the 200^{th} generation in a clearer way, we show the cutting planes of Fig. 6-C(b-1) in Appendix B.

We discover three interesting patterns based on Fig. 6.

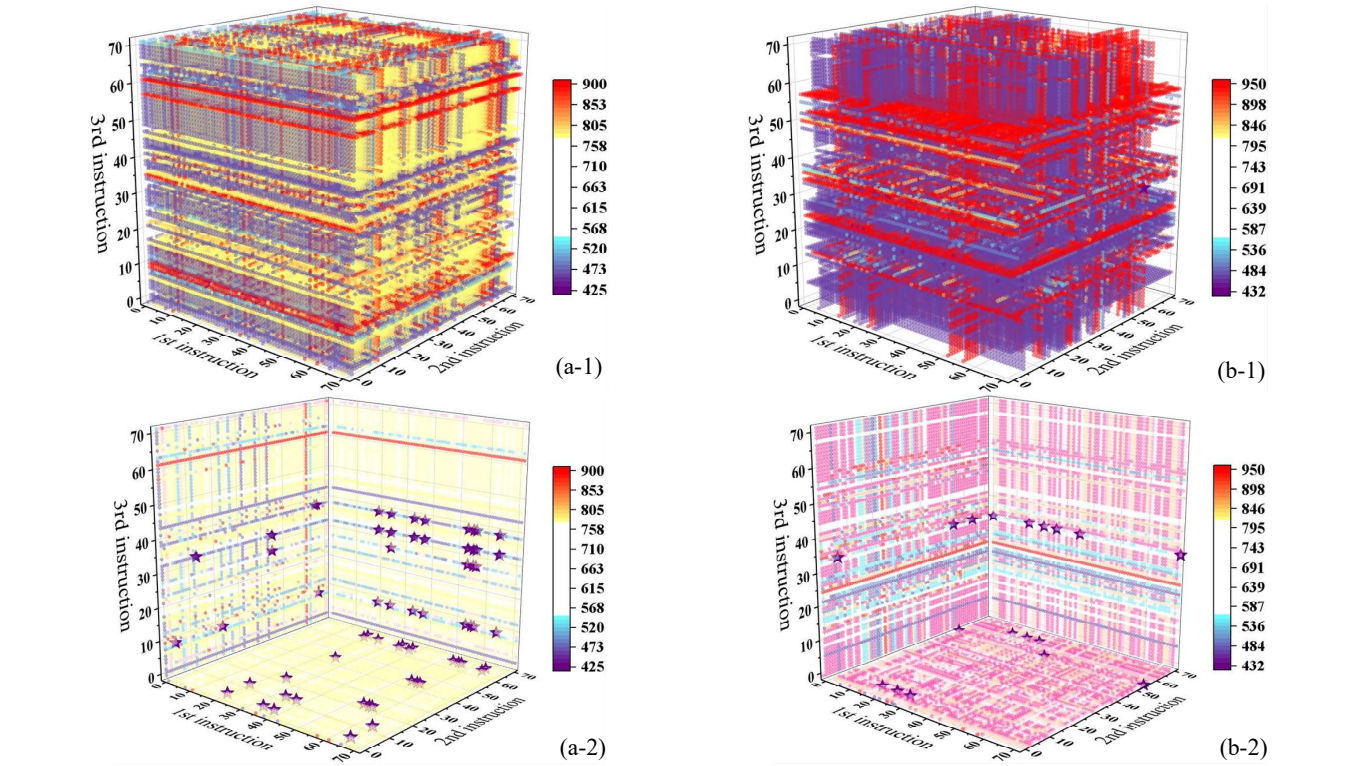
1) Aggregating Solutions with Different Fitness: Fig. 6 confirms the optimization on FLs. In the three problems, good and bad solutions are distributed uniformly across the initial FLs. For example, the blue lines in Fig. 6-B(a) spread across the space. However, we can see that the good and bad solutions aggregate respectively during the optimization. For example, the good solutions aggregate to the bottom-left corner in Fig. 6-A(c), the good solutions aggregate to the top-right corner in Fig. 6-B(c), and the good solutions approximately aggregate to the indices from 15 to 35 on 2^{nd}-3^{rd}-instruction plane in Fig. 6-C(b-2). More interestingly, for simple problems (i.e., Finger and Toy) whose optimal solutions are likely found by LGP, the optimal solutions are located near the diagonal after FLO, so that they are close to other good solutions. For



A. FLs of Finger: (a) initial FL, (b) FL at the 10th generation, (c) FL at the 200th generation



B. FLs of Toy: (a) initial FL, (b) FL at the 10th generation, (c) FL at the 200th generation



C. FLs of <Tmean, 0.85>: (a-1) and (a-2) initial FL, (b-1) and (b-2) FL at the 200th generation. (a-2) and (b-2) are the FLs projected on x-y, x-z, and y-z planes.

Fig. 6. The example FLs of Finger, Toy, and ⟨Tmean,0.85⟩. The cool tone indicates good fitness and the warm tone indicates bad fitness. The purple stars indicate the optimal solutions. The instructions at the bottom of A are the corresponding instructions for each index.

more difficult problems such as the DJSS problem ⟨Tmean, 0.85⟩, the optimal solutions are unnecessarily located near the diagonal, but they are located near the cold-tone regions.

2) Fitness Aligning: We can see a *fitness aligning* pattern when the number of possible introns in solutions[3] increases (i.e., Fig. 6-B and -C). The fitness aligning means that the solutions with very similar fitness allocate along an axis or a hyperplane (i.e., the fitness is aligned), rather than aggregating as an ellipse or a hypersphere. For example, the solutions with the same color in Fig. 6-B(a) allocate along axes, and there are considerable 2-D planes with the same color across Fig. 6-C(b-1). When there are a large number of introns in the solutions, a solution can easily reach another solution with the same fitness by adding, removing, or modifying introns, which is equivalent to moving along axes or against hyperplanes. The fitness aligning pattern shows the important role of introns in performing neutral moves. It is consistent with an empirical conclusion in existing LGP studies that introns in LGP problems are effective in reducing destructive search [53], [66].

3) Diagonal Symmetry: Fig. 6 implies a *diagonal symmetry* pattern of FLs. The diagonal symmetry means that a fitness landscape of LGP is symmetrical diagonally approximately. For example, we can see two red lines 1^{st}-instruction$= 105$ and 2^{nd}-instruction$= 105$ in Fig. 6-B(a). Although the red line 2^{nd}-instruction$= 105$ is more "solid" than 1^{st}-instruction$= 105$, they are nearly symmetrical diagonally overall. This implies that an instruction likely leads to similar effectiveness when it is placed on both the first and second instructions. The diagonal symmetry can also be seen roughly in the 2^{nd}-3^{rd}-instruction plane in Fig. 6-C(b-2). Because the instructions likely have similar effectiveness in similar places of LGP programs, performing crossover to share effective instructions is an important method to search for better solutions in existing LGP studies [53].

Furthermore, Fig. 6-C(b-2) shows that the diagonal symmetry is dependent on two consecutive instructions (i.e., a clearer visualization is shown in Appendix B). We can see the diagonal symmetry on the 2^{nd}-3^{rd}-instruction plane but not on the 1^{st}-3^{rd}-instruction plane. Given the importance of neutral moves in LGP search [53] and the connection between fitness aligning and mutating only one instruction, we highly suspect that the diagonal symmetry implies a missed operator in existing LGP studies, that is, swapping two consecutive instructions. Swapping two consecutive instructions makes use of the diagonal symmetry to perform neutral moves on instruction positions in an LGP solution. Based on the visualized FLs, Table IV shows the relationships among the two newly found patterns, the LGP operators, and the (non-)neutral move on instructions (i.e., what instructions should be used) and instruction positions (i.e., where should place the instruction).

[3]An intron is an LGP instruction that does not contribute to the final output of an LGP program. An exon is an LGP instruction that contributes to the final output of an LGP program.

TABLE IV
VISUALIZED EXPLANATIONS ON (NON-)NEUTRAL MOVE OF INSTRUCTIONS AND THEIR POSITIONS

Patterns	Search Operators	Variation on LGP Programs
Fitness aligning	Mutating introns	neutral move on instructions
	Mutating exons	non-neutral move on instructions
Diagonal symmetry	Swapping consecutive instructions	neutral move on instruction positions
	Swapping inconsecutive instructions	non-neutral move on instruction positions

VI. FURTHER ANALYSIS

A. Test Performance on common Benchmarks

The previous section has verified that the proposed method can effectively reduce the hardness of FLs. This section further investigates whether the proposed FLO method is effective in larger search spaces with higher dimensions and helps the LGP method achieve better performance. To answer this question, this section applies LGP with FLO to search symbolic solutions for common benchmarks. Specifically, FLO optimizes the FL during LGP evolution, and LGP searches for new solutions based on the up-to-date optimized FLs. Since the running overhead of FLO is correlated to the number of instructions, we limit each LGP instruction to having up to one operation and one input feature to reduce the number of instructions.

We select ten benchmark problems from symbolic regression and DJSS, as shown in Table V. Nguyen4 has a lot of repetitive multiplications and additions, which is aligned with the main assumption of frequency-based mutation, that is, the frequency of primitives implies the importance. In contrast, Keijzer11 and R1 have much less repetitive patterns. Airfoil, BHouse, and Redwine are three typical real-world regression benchmarks downloaded from the UCI machine learning datasets. We apply relative square error (RSE) to measure performance in symbolic regression. ⟨Tmax, 0.95⟩ to ⟨WFmean, 0.95⟩ are four DJSS problems with different optimization objectives for the overall job shop (i.e., maximum tardiness, mean tardiness, maximum flowtime, and weighted mean flowtime). A utilization level of 0.95 indicates a much busier and more complex job shop than a utilization level of 0.85.

To verify the effectiveness of the proposed method, we compare four methods. First, basic LGP serves as a baseline, denoted as "basicLGP". The second method is a basic LGP method that applies frequency-based mutation (i.e., freqmut). The third method is an LGP that applies an operator swapping consecutive instructions. The third method verifies our insight into the possibility of neutral move of instruction positions, denoted by "swap". The fourth method is an advanced method (denoted as ADVAN). ADVAN represents different methods in symbolic regression and DJSS problems; these are semantic LGP [64] and grammar-guided LGP [67] for these two domains, respectively. ADVAN parameters are set following their recommendation. The final method is the one proposed here which automatically optimizes the FL and applies basic LGP

TABLE V
THE TEN BENCHMARK PROBLEMS

Problems	Formula
Synthetic symbolic regression benchmarks	
Nguyen4	$f(x_0) = x_0^6 + x_0^5 + x_0^4 + x_0^3 + x_0^2 + x_0$
Keijzer11	$f(x_0, x_1) = x_0 x_1 + \sin((x_0 - 1)(x_1 - 1))$
R1	$f(x_0) = \frac{(x_0+1)^3}{x_0^2 - x_0 + 1}$
Real-world symbolic regression benchmarks	
Airfoil	unknown
BHouse	unknown
Redwine	unknown
DJSS benchmarks	
$\langle$Tmax, 0.95$\rangle$	unknown
$\langle$Tmean, 0.95$\rangle$	unknown
$\langle$Fmax, 0.95$\rangle$	unknown
$\langle$WFmean, 0.95$\rangle$	unknown

to search against the optimized FL (denoted as LGP-FLO). Specifically, LGP-FLO searches against the optimized FL by moving a symbolic solution toward another better one based on index vectors. The index vector of the symbolic solution moves within its neighborhood, reaches another index vector, and rebuilds a new symbolic solution. LGP-FLO also uses neutral moves on instructions and their positions during the search. The parameters of FLO (i.e., B and σ) are set as 10 and 0.1, respectively. The threshold of neighborhood structures ϵ is approximately 5% of the total number of instructions, that is, 100 for symbolic regression problems and 1000 for DJSS problems. The other parameters of basic LGP follow [53], [54].

All methods compared have the same primitive sets for the same problems. The function set for symbolic regression is $\{+, -, \times, \div, \sin, \cos, \sqrt{|\cdot|}, \ln(|\cdot|)\}$. The terminal set for symbolic regression is the problem input features. The function set for DJSS is $\{+, -, \times, \div, \max, \min\}$ (ADVAN method in DJSS includes "IF", as recommended). There are thirteen input features in the DJSS problems, including the processing time of operations and the number of remaining operations in machine queues [54]. All methods evolve a population of 256 individuals for 200 generations to search for the fittest solutions.

Table VI shows mean test performance (and standard deviation) over 50 independent runs. A Friedman test on the compared methods shows a p-value of 1.7e-4, which indicates a significant difference in performance. We further apply a Wilcoxon rank-sum test with Bonferroni correction and a significance level of 0.05 to analyze the performance of compared methods versus LGP-FLO on each benchmark. The notations of "+", "−", "≈" are the same as in Table III.

We can see that by optimizing the FL over the search, we can significantly improve the performance of stochastic symbolic search methods. Specifically, LGP-FLO has significantly better performance than basic LGP in many benchmarks and LGP with advanced manually-designed operators (i.e., freqmut). Besides, LGP-FLO also has a very competitive performance with ADVAN methods in both supervised learning and dynamic combinatorial optimization problems. The comparison between LGP-FLO and ADVAN confirms that searching for automatically optimized FLs is very competitive to those manually-designed ADVAN methods. The mean ranks of the Friedman test show that LGP-FLO has the best overall performance (i.e., 1.75) amongst the compared methods on the tested benchmark problems. These results imply that FLO is effective in higher dimensions and larger search spaces, given that these benchmarks have a large primitive set and require long programs.

By comparing "swap" and "basicLGP", we demonstrate a significant performance improvement with "swap". We confirm that the neutral move of instruction positions is an effective operator for LGP search. Given that swapping consecutive instructions to perform a neutral move of instruction positions is a newly discovered operator based on the pattern analysis of optimized FLs, we believe that the optimized FLs and their pattern analysis help explore more effective and explainable search mechanisms for stochastic symbolic search.

B. Parameter Sensitivity

FLO has two main parameters, the number of sampled solutions from the population B and the maximum step size σ. They are set as 10 and 0.1 by default. This section investigates the sensitivity of these two parameters. Specifically, we compare the FL hardness of $B = 5, 10, 15, 20$ and $\sigma = 0.005, 0.05, 0.1, 0.5, 1$, and apply a Friedman test with significance level of 0.05 to analyze overall performance on tested problems (i.e., Finger, Toy, R1, and $\langle$Tmean, 0.85$\rangle$), as shown in Tables VII and VIII. Other parameter settings are kept the same as in section IV-D.

We can see that all the p-values in Tables VII and VIII are larger than 0.05, indicating that there is no significant difference among the choices of B and σ over the four FL metrics. These results confirm that B and σ are relatively robust parameters.

VII. CONCLUSIONS

The results imply an encouraging potential of the proposed FLO method to find better or very competitive landscapes automatically. The proposed FLO method is essentially an optimization approach that optimizes the symbol indices of symbols to construct better neighborhood structures for symbolic solutions. Specifically, FLO aggregates good solutions, separates good solutions from bad ones, and encourages the new neighborhood structures to be consistent with the domain knowledge. We apply LGP, an evolutionary computation-based stochastic symbolic search method, to investigate the effectiveness of the proposed FLO. Based on the results, four main conclusions are drawn:

1) Evaluating four kinds of FL metrics reveals that FLO successfully finds significantly more cone-like landscapes than manually designed landscapes. The cone-like landscapes are also confirmed by the visualized landscape after optimization.
2) Visualization results on LGP landscapes show two important patterns of LGP landscapes. If there are LGP introns in the search space, solutions with similar fitness likely aggregate to a hyperplane on FL (i.e., fitness aligning), and the hyperplanes spanned by two consecutive

TABLE VI
MEAN TEST PERFORMANCE (AND STANDARD DEVIATION) ON THE TEN BENCHMARK PROBLEMS. THE BEST MEAN PERFORMANCE IS HIGHLIGHTED IN BOLD.

Problems	basicLGP	freqmut	swap	ADVAN	LGP-FLO (Ours)
Nguyen4	0.149 (0.248) $-$	0.069 (0.052) $\approx$	0.071 (0.064) $-$	0.052 (0.064) $\approx$	**0.048 (0.045)**
Keijzer11	0.339 (0.142) $-$	0.299 (0.103) $-$	0.288 (0.121) $-$	**0.213 (0.091)** $\approx$	0.22 (0.104)
R1	0.034 (0.035) $-$	0.02 (0.026) $-$	0.016 (0.027) $-$	0.011 (0.034) $-$	**0.01 (0.02)**
Airfoil	0.643 (0.132) $-$	0.557 (0.113) $\approx$	0.559 (0.114) $\approx$	0.524 (0.049) $\approx$	**0.521 (0.095)**
Bhouse	0.404 (0.126) $\approx$	0.443 (0.11) $\approx$	0.374 (0.114) $+$	**0.312 (0.063)** $+$	0.417 (0.088)
Redwine	0.759 (0.034) $-$	0.74 (0.037) $\approx$	0.741 (0.033) $\approx$	**0.699 (0.025)** $+$	0.735 (0.031)
$\langle$Tmax,0.95$\rangle$	3999.2 (90.9) $\approx$	3976.3 (190.1) $\approx$	3969.1 (91.5) $\approx$	3968.8 (112.3) $\approx$	**3967.5 (88.3)**
$\langle$Tmean,0.95$\rangle$	1118.2 (10.7) $\approx$	1114.1 (9.1) $\approx$	**1113.8 (11.4)** $\approx$	1116.3 (12.1) $\approx$	1115 (7.7)
$\langle$Fmax,0.95$\rangle$	4585.4 (126.1) $-$	4523.2 (107.3) $\approx$	4518.4 (70.3) $-$	4501 (74.8) $\approx$	**4477.4 (71.4)**
$\langle$WFmean,0.95$\rangle$	2715.8 (16.4) $-$	2710.6 (37.7) $\approx$	**2708.8 (20.6)** $\approx$	2711.7 (21.5) $\approx$	2711.6 (26.4)
mean ranks	4.8	3.3	2.95	2.2	**1.75**
pair-wise p-value	1.4E-4	0.276	0.881	1	

TABLE VII
FRIEDMAN TEST OVER B ON THE FOUR METRICS

metrics		$B=5$	$B=10$	$B=15$	$B=20$
FDC	mean rank	3.25	**2**	2.25	2.5
	p-value		0.552		
NSC	mean rank	**2.25**	**2.25**	2.5	2.5
	p-value		0.96		
RBS	mean rank	**1.25**	2.25	3.5	3
	p-value		0.075		
EVO	mean rank	**2**	2.25	3.25	2.5
	p-value		0.552		
mean rank over metrics		2.188	2.188	2.875	2.625

TABLE VIII
FRIEDMAN TEST OVER σ ON THE FOUR METRICS

metrics		$\sigma=0.005$	$\sigma=0.05$	$\sigma=0.1$	$\sigma=0.5$	$\sigma=1$
FDC	mean rank	3.25	3	**2.75**	3	3
	p-value			0.995		
NSC	mean rank	3.25	3	**2.75**	3.25	**2.75**
	p-value			0.981		
RBS	mean rank	**2**	**2**	2.75	4.75	3.5
	p-value			0.060		
EVO	mean rank	2.5	3.5	3.5	**2.25**	3.25
	p-value			0.678		
mean rank over metrics		2.75	2.875	2.938	3.313	3.125

LGP instruction positions normally have a diagonally symmetric layout (i.e., diagonal symmetry).

3) Pattern analysis on visualized FLs further helps to design a new operator for LGP, swapping two consecutive instructions, which is missed by existing LGP studies. The results on common benchmarks confirm that swapping consecutive instructions implements an essential capability of fine-tuning instruction positions, which improves LGP performance.
4) LGP that searches against the optimized FLs achieves very competitive performance with advanced methods within the same number of fitness evaluations. This implies that the automatically designed landscapes achieve competitive effects to those designed by human experts.

To the best of our knowledge, this paper is the first attempt to explicitly optimize FLs of stochastic symbolic search automatically. The proposed FLO method is general enough to be applied to other domains (e.g., classification) and to other stochastic symbolic search methods once their solutions are represented as a list of symbols (e.g., a tree-based GP program represents as a symbol list by preorder traversal). We will investigate the effectiveness of the proposed FLO method in other domains and with other symbolic search methods in the future. Our experiments use four common FL metrics to evaluate the quality of FLs. This also facilitates further investigations of the correlation of these FL metrics, missed by existing FL analysis studies. In future work, we will further improve the search mechanisms (e.g., adaptive local search) over the optimized FLs to enhance the effectiveness. Note that since the FLO running overhead increases with the primitive set, we need to further reduce the overhead in the future.

REFERENCES

[1] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 550–570, 2023.

[2] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, X. Ning, Y. Ma, H. Yang, B. Yu, H. Yang, and Y. Wang, "Machine learning for electronic design automation: A survey," *ACM Transactions on Design Automation of Electronic Systems*, vol. 26, no. 5, pp. 1–46, 2021.

[3] R. Alur, R. Singh, D. Fisman, and A. Solar-Lezama, "Search-based program synthesis," *Communications of the ACM*, vol. 61, no. 12, pp. 84–93, 2018.

[4] K. Kinnear, "Fitness landscapes and difficulty in genetic programming," in *Proceedings of the First IEEE Conference on Evolutionary Computation and IEEE World Congress on Computational Intelligence*, 1994, pp. 142–147.

[5] N. M. Rodrigues, K. M. Malan, G. Ochoa, L. Vanneschi, and S. Silva, "Fitness landscape analysis of convolutional neural network architectures for image classification," *Information Sciences*, vol. 609, pp. 711–726, 2022.

[6] L. Vanneschi, "Fitness landscapes and problem hardness in genetic programming," in *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, 2010, pp. 2711–2738.

[7] J. R. Koza, *Genetic Programming : On the Programming of Computers By Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[8] 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.

[9] C. Raymond, Q. Chen, B. Xue, and M. Zhang, "Learning symbolic model-agnostic loss functions via meta-learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 699–13 714, 2023.

[10] Q. Fan, Y. Bi, B. Xue, and M. Zhang, "A genetic programming-based method for image classification with small training data," *Knowledge-Based Systems*, vol. 283, p. 111188, 2024.

[11] N. Haut, W. Banzhaf, B. Punch, and D. Colbry, "Accelerating Image Analysis Research with Active Learning Techniques in Genetic Programming," in *Genetic Programming Theory and Practice XX*, ser. Genetic and Evolutionary Computation. Singapore: Springer Nature, 2024, pp. 45–64.

[12] Q. Chen, B. Xue, W. Browne, and M. Zhang, "Evolutionary regression and modelling," in *Handbook of Evolutionary Machine Learning*. Singapore: Springer Nature Singapore, 2024, pp. 121–149.

[13] N. Makke and S. Chawla, "Interpretable scientific discovery with symbolic regression: a review," *Artificial Intelligence Review*, vol. 57, no. 1, p. 2, 2024.

[14] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, *Genetic Programming for Production Scheduling*. Singapore: Springer Singapore, 2021.

[15] Z. Huang, Y. Mei, F. Zhang, and M. Zhang, "Multitask Linear Genetic Programming with Shared Individuals and its Application to Dynamic Job Shop Scheduling," *IEEE Transactions on Evolutionary Computation*, pp. 1–15, 2023.

[16] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[17] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Guided Subtree Selection for Genetic Operators in Genetic Programming for Dynamic Flexible Job Shop Scheduling," in *Proceedings of European Conference on Genetic Programming*, 2020, pp. 262–278.

[18] M. Wan, T. Weise, and K. Tang, "Novel loop structures and the evolution of mathematical algorithms," in *Proceedings of European Conference on Genetic Programming*, 2011, pp. 49–60.

[19] T. Jones and S. Forrest, "Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms," in *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, pp. 184–192.

[20] L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue, "A Survey of Problem Difficulty in Genetic Programming," in *Proceedings of the Congress of the Italian Association for Artificial Intelligence*, 2005, pp. 66–77.

[21] M. Clergue, P. Collard, M. Tomassini, and L. Vanneschi, "Fitness Distance Correlation And Problem Difficulty For Genetic Programming." in *Proceedings of Genetic and Evolutionary Computation Conference*, 2002, pp. 724–732.

[22] L. Vanneschi, M. Tomassini, M. Clergue, and P. Collard, "Difficulty of Unimodal and Multimodal Landscapes in Genetic Programming," in *Genetic and Evolutionary Computation*, 2003, pp. 1788–1799.

[23] L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue, "Fitness Distance Correlation in Structural Mutation Genetic Programming," in *Proceedings of European Conference on Genetic Programming*, 2003, pp. 455–464.

[24] L. Vanneschi and M. Tomassini, "Pros and Cons of Fitness Distance Correlation in Genetic Programming," in *Proceedings of Genetic and Evolutionary Computation Conference Workshop Program*, 2003, pp. 284–287.

[25] L. Vanneschi, "Theory and practice for efficient genetic programming," phdthesis, Université de Lausanne, Faculté des sciences, 2004.

[26] R. Čorić, M. Đumić, and D. Jakobović, "Genetic programming hyperheuristic parameter configuration using fitness landscape analysis," *Applied Intelligence*, vol. 51, no. 10, pp. 7402–7426, 2021.

[27] G. Ochoa, M. Tomassini, S. Vérel, and C. Darabos, "A study of NK landscapes' basins and local optima networks," in *Proceedings of the Genetic and evolutionary computation Conference*, 2008, pp. 555–562.

[28] G. Ochoa, S. Verel, F. Daolio, and M. Tomassini, "Local Optima Networks: A New Model of Combinatorial Fitness Landscapes," in *Recent Advances in the Theory and Application of Fitness Landscapes*, 2014, vol. 6, pp. 233–262.

[29] M. Durasevic, D. Jakobovic, M. Scoczynski Ribeiro Martins, S. Picek, and M. Wagner, "Fitness Landscape Analysis of Dimensionally-Aware Genetic Programming Featuring Feynman Equations," in *Proceedings of International Conference on Parallel Problem Solving from Nature*, 2020, pp. 111–124.

[30] Y. He and F. Neri, "Fitness Landscape Analysis of Genetic Programming Search Spaces with Local Optima Networks," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 2056–2063.

[31] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue, "A Study of Fitness Distance Correlation as a Difficulty Measure in Genetic Programming," *Evolutionary Computation*, vol. 13, no. 2, pp. 213–239, 2005.

[32] L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue, "Fitness distance correlation in genetic programming: a constructive counterexample," in *Proceedings of the Congress on Evolutionary Computation*, 2003, pp. 289–296.

[33] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Vérel, "Fitness Clouds and Problem Hardness in Genetic Programming," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2004, pp. 690–701.

[34] S. Verel, P. Collard, and M. Clergue, "Where are bottlenecks in NK fitness landscapes?" in *Proceedings of The 2003 Congress on Evolutionary Computation*, 2003, pp. 273–280.

[35] L. Vanneschi, M. Tomassini, P. Collard, and S. Vérel, "Negative Slope Coefficient: A Measure to Characterize Genetic Programming Fitness Landscapes," in *Proceedings of European Conference on Genetic Programming*, 2006, pp. 178–189.

[36] T. Hu, J. L. Payne, W. Banzhaf, and J. H. Moore, "Robustness, evolvability, and accessibility in linear genetic programming," in *European Conference on Genetic Programming*, 2011, pp. 13–24.

[37] T. Hu, J. L. Payne, W. Banzhaf, and J. H. Moore, "Evolutionary dynamics on multiple scales: A quantitative analysis of the interplay between genotype, phenotype, and fitness in linear genetic programming," *Genetic Programming and Evolvable Machines*, vol. 13, no. 3, pp. 305–337, 2012.

[38] T. Hu, W. Banzhaf, and J. H. Moore, "Robustness and evolvability of recombination in linear genetic programming," in *European Conference on Genetic Programming*, vol. 7831, 2013, pp. 97–108.

[39] T. Hu and W. Banzhaf, "Neutrality, Robustness, and Evolvability in Genetic Programming," in *Genetic Programming Theory and Practice XIV*, 2018, pp. 101–117.

[40] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon, "Defining locality as a problem difficulty measure in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 12, no. 4, pp. 365–401, 2011.

[41] F. Rothlauf and M. Oetzel, "On the Locality of Grammatical Evolution," in *Proceedings of European Conference on Genetic Programming*, 2006, pp. 320–330.

[42] K. Slaný and L. Sekanina, "Fitness Landscape Analysis and Image Filter Evolution Using Functional-Level CGP," in *Proceedings of European Conference on Genetic Programming*, 2007, pp. 311–320.

[43] N. Haut, W. Banzhaf, and B. Punch, "Correlation Versus RMSE Loss Functions in Symbolic Regression Tasks," in *Genetic Programming Theory and Practice XIX*, ser. Genetic and Evolutionary Computation. Singapore: Springer Nature, 2023, pp. 31–55.

[44] Q. Chen, B. Xue, and W. Banzhaf, "Relieving Coefficient Learning in Genetic Programming for Symbolic Regression via Correlation and Linear Scaling," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2023, pp. 420–428.

[45] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Multitask Multiobjective Genetic Programming for Automated Scheduling Heuristic Learning in Dynamic Flexible Job-Shop Scheduling," *IEEE Transactions on Cybernetics*, pp. 1–14, 2022.

[46] T. Wei and J. Zhong, "A Preliminary Study of Knowledge Transfer in Multi-Classification Using Gene Expression Programming," *Frontiers in Neuroscience*, vol. 13, no. January, pp. 1–14, 2020.

[47] Y. Bi, B. Xue, and M. Zhang, "Learning and Sharing: A Multitask Genetic Programming Approach to Image Feature Learning," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 2, pp. 218–232, 2022.

[48] T. Helmuth, L. Spector, and J. Matheson, "Solving Uncompromising Problems With Lexicase Selection," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 630–643, 2015.

[49] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "A Double Lexicase Selection Operator for Bloat Control in Evolutionary Feature Construction for Regression," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Jul. 2023, pp. 1194–1202.

[50] J. F. Miller, "An empirical study of the efficiency of learning boolean functions using a Cartesian Genetic Programming approach," *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, no. December, pp. 1135–1142, 1999.

[51] C. Ferreira, "Gene Expression Programming: a New Adaptive Algorithm for Solving Problems," *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[52] L. F. D. P. Sotto, P. Kaufmann, T. Atkinson, R. Kalkreuth, and M. Porto Basgalupp, "Graph representations in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 22, no. 4, pp. 607–636, 2021.

[53] M. Brameier and W. Banzhaf, *Linear Genetic Programming*. Springer New York, NY, 2007.

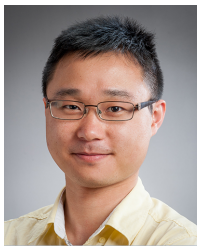
[54] Z. Huang, Y. Mei, F. Zhang, and M. Zhang, "A Further Investigation to Improve Linear Genetic Programming in Dynamic Job Shop Scheduling," in *Proceedings of IEEE Symposium Series on Computational Intelligence*, Dec. 2022, pp. 496–503.

[55] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving Scheduling Heuristics via Genetic Programming with Feature Selection in Dynamic

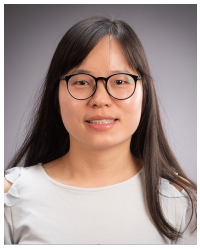
Flexible Job-Shop Scheduling," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1797–1811, 2021.
[56] S. Forstenlechner, D. Fagan, M. Nicolau, and M. O'Neill, "Extending Program Synthesis Grammars for Grammar-Guided Genetic Programming," in *Parallel Problem Solving from Nature – PPSN XV*, vol. 11101, 2018, pp. 197–208.
[57] C. S. Pereira, D. M. Dias, M. A. C. Pacheco, M. M. Vellasco, A. V. Abs Da Cruz, and E. H. Hollmann, "Quantum-Inspired Genetic Programming Algorithm for the Crude Oil Scheduling of a Real-World Refinery," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3926–3937, 2020.
[58] Z. Huang, Y. Mei, F. Zhang, and M. Zhang, "Grammar-guided Linear Genetic Programming for Dynamic Job Shop Scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, NY, USA, Jul. 2023, pp. 1137–1145.
[59] L. F. D. P. Sotto and V. V. d. Melo, "Studying bloat control and maintenance of effective code in linear genetic programming for symbolic regression," *Neurocomputing*, vol. 180, pp. 79–93, 2016.
[60] T. Hu, G. Ochoa, and W. Banzhaf, "Phenotype Search Trajectory Networks for Linear Genetic Programming," in *Proceedings of European Conference on Genetic Programming*, vol. 13986 LNCS, 2023, pp. 52–67.
[61] Z. Huang, Y. Mei, F. Zhang, and M. Zhang, "Graph-based linear genetic programming: a case study of dynamic scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, NY, USA, Jul. 2022, pp. 955–963.
[62] T. P. Pawlak, B. Wieloch, and K. Krawiec, "Semantic Backpropagation for Designing Search Operators in Genetic Programming," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 326–340, 2015.
[63] T. P. Pawlak and K. Krawiec, "Competent Geometric Semantic Genetic Programming for Symbolic Regression and Boolean Function Synthesis," *Evolutionary Computation*, vol. 26, no. 2, pp. 177–212, 2018.
[64] Z. Huang, Y. Mei, and J. Zhong, "Semantic Linear Genetic Programming for Symbolic Regression," *IEEE Transactions on Cybernetics*, pp. 1–14, 2022.
[65] J. Zhong, Y.-S. Ong, and W. Cai, "Self-Learning Gene Expression Programming," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 65–80, 2016.
[66] L. F. D. P. Sotto, F. Rothlauf, V. V. de Melo, and M. P. Basgalupp, "An analysis of the influence of noneffective instructions in linear genetic programming," *Evolutionary Computation*, vol. 30, no. 1, pp. 51–74, 2022.
[67] Z. Huang, Y. Mei, F. Zhang, and M. Zhang, "Toward Evolving Dispatching Rules With Flow Control Operations By Grammar-Guided Linear Genetic Programming," *IEEE Transactions on Evolutionary Computation*, pp. 1–15, 2024.

Zhixing Huang received the B.S. and M.S. degrees from the School of Computer Science and Engineering, South China University of Technology, China, in 2018 and 2020, respectively, and received his Ph.D degree from the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand, in 2024. He is currently a postdoctoral fellow in Artificial Intelligence with the Centre of Data Science and AI & the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. His research interests include evolutionary computation (e.g., genetic programming and its applications), combinatorial optimization, and evolutionary learning.

Dr. Huang is a member of the IEEE Computational Intelligence Society and IEEE Taskforce on Evolutionary Scheduling and Combinatorial Optimisation. He has been serving as a reviewer for top international journals such as the IEEE Transactions on Evolutionary Computation and the IEEE Transactions on Neural Networks and Learning Systems, and conferences including the Genetic and Evolutionary Computation Conference and the IEEE Congress on Evolutionary Computation.



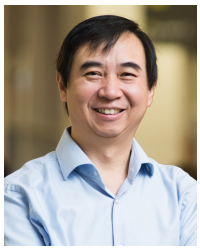
Yi Mei is an Associate Professor at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. His research interests include evolutionary computation and machine learning for combinatorial optimisation, genetic programming, hyper-heuristics, and explainable AI. He is an Associate Editor of IEEE Transactions on Evolutionary Computation, IEEE Transactions on Artificial Intelligence, and Journal of Scheduling. He is the Chair of IEEE Taskforce on Evolutionary Scheduling and Combinatorial Optimisation. He is a Fellow of Engineering New Zealand and an IEEE Senior Member.



Fangfang Zhang received the B.Sc. and M.Sc. degrees from Shenzhen University, China, and the PhD degree in Computer Science from Victoria University of Wellington, New Zealand, in 2014, 2017, and 2021, respectively.

She is currently a lecturer in the Centre for Data Science and Artifical Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. She has more than 65 publications in refereed international journals and conferences. Her research interests include evolutionary computation, hyper-heuristic learning/optimisation, job shop scheduling, surrogate, and multitask learning.

Dr Fangfang is an Associate Editor of Expert Systems With Applications, and Swarm and Evolutionary Computation. She is a Vice-Chair of the Task Force on Evolutionary Scheduling and Combinatorial Optimisation. She is a member of the IEEE Computational Intelligence Society and Association for Computing Machinery, and has been severing as reviewers for top international journals and conferences. She is the Secretary of IEEE New Zealand Central Section, and was the Chair of the IEEE Student Branch at Victoria University of New Zealand and the chair of Professional Activities Coordinator.



Mengjie Zhang received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively.

He is currently a Professor of Computer Science, the Head of the Evolutionary Computation and Machine Learning Research Group, and the Associate Dean (Research and Innovation) with the Faculty of Engineering, Victoria University of Wellington, New Zealand. His current research interests include genetic programming, image analysis, feature selection and reduction, job-shop scheduling, and evolutionary deep learning and transfer learning. He has published over 800 research papers in refereed international journals and conferences. He is a Fellow of the Royal Society of New Zealand, a Fellow of Engineering New Zealand, a Fellow of IEEE, and an IEEE Distinguished Lecturer.

Wolfgang Banzhaf received the Dr.rer.nat (Ph.D.) degree from the Department of Physics, Technische Hochschule Karlsruhe (currently, Karlsruhe Institute of Technology), Karlsruhe, Germany, in 1985. He was the University Research Professor with the Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada, where he served as the Head of Department from 2003 to 2009 and from 2012 to 2016. He is the John R. Koza Chair of Genetic Programming with the Department of Computer Science and Engineering and a member of the BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI, USA. Studies of self-organization and the field of Artificial Life are also of very much interest to him. He has become more involved with network research as it applies to natural and man-made systems. His research interests are in the field of bioinspired computing, notably evolutionary computation, and complex adaptive systems.