# Learning a Data-Driven Beam Search Heuristic Using a Feed-Forward Neural Network and Genetic Algorithms

## 1 Overview

This work proposes a learning framework for constructing a data-driven heuristic to guide Beam Search in combinatorial optimization problems. A feed-forward neural network (FFNN) is used to parameterize the heuristic function. Instead of relying on gradient-based training, the network weights are optimized using a Genetic Algorithm (GA).

Each individual in the GA population represents a complete set of neural network weights. The quality of an individual is evaluated by embedding the neural network inside Beam Search and measuring the average solution quality obtained on a set of training instances. Validation instances are used exclusively to monitor generalization.

Formally, the learning problem is defined as:

$$\max_{\theta} \frac{1}{|T|} \sum_{I \in T} f(\text{BeamSearch}(I, h_\theta)),$$

where $T$ denotes the training set, $h_\theta$ is the neural heuristic parameterized by weights $\theta$, and $f(\cdot)$ measures solution quality.

## 2 Neural Network Model

The heuristic is represented by a multilayer perceptron with an arbitrary number of hidden layers and user-defined units per layer. Supported activation functions include hyperbolic tangent, ReLU, and sigmoid.

For each layer $l$, forward propagation is defined as:

$$\mathbf{z}^{(l+1)} = W^{(l)}\mathbf{a}^{(l)} + b^{(l)}, \quad \mathbf{a}^{(l+1)} = \phi(\mathbf{z}^{(l+1)}),$$

where $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector, respectively, and $\phi$ denotes the activation function.

All weights and biases are flattened into a single vector, which forms the chromosome of a GA individual.

## 3 Fitness Evaluation

Given a candidate weight vector:

1. The weights are loaded into the FFNN.
2. Beam Search is executed on each training instance using the neural heuristic.
3. The size of the best sequence obtained for each instance is recorded.

The fitness value is computed as the average solution quality over all training instances. Validation performance is computed analogously on a separate dataset and is used only for reporting.

## 4 Genetic Algorithm

The optimization process follows a population-based evolutionary scheme.

### 4.1 Initialization

Each individual is initialized by sampling weights uniformly from a predefined interval $[-w, w]$.

### 4.2 Elitism

A fixed number of top-performing individuals are copied unchanged into the next generation.

### 4.3 Mutation

Several individuals are generated completely at random (random immigrants). This mechanism promotes exploration and maintains population diversity.

### 4.4 Offspring Generation

Three alternative parent-selection strategies are supported:

- **RKGA**: Two parents are selected uniformly at random, and each gene is inherited from either parent with probability 0.5.
- **BRKGA**: One parent is selected from the elite set and one from the non-elite set. Each gene is inherited from the elite parent with a predefined probability.
- **Lexicase Selection**: Training instances are randomly shuffled. Individuals are filtered iteratively by retaining those achieving the best performance on each instance. Parents are selected randomly from the remaining candidates.

### 4.5 Replacement and Termination

The new population consists of elites, mutants, and offspring. The algorithm terminates once a predefined time limit is reached. The best individual encountered during training is continuously tracked and stored.

## 5 Overall Learning Procedure

The learning process alternates between evolutionary optimization of network weights and evaluation through (iterative multi-source) Beam Search. Importantly, the neural network does not directly predict solutions; instead, it learns how to guide the search procedure. This places the method within the class of neuro-evolutionary hyper-heuristics.

## 6 Algorithm

**Algorithm 1** GA-Based Learning of a Beam Search Heuristic

```
1: Initialize population $P$ with random neural network weights
2: Evaluate all individuals on training set $T$
3: Store best solution $\theta^*$
4: $start \leftarrow$ current time
5: while current time $-start < T_{\max}$ do
6:     Sort $P$ by fitness (descending)
7:     Initialize empty population $P'$
8:     Copy top $n_{elites}$ individuals from $P$ to $P'$
9:     for $i = 1$ to $n_{mutants}$ do
10:        Create random individual $x$
11:        Evaluate $x$ on $T$
12:        Add $x$ to $P'$
13:        Update $\theta^*$ if improved
14:    end for
15:    for $i = 1$ to $n_{offspring}$ do
16:        Select parents using RKGA, BRKGA, or Lexicase
17:        Generate offspring by crossover
18:        Evaluate offspring on $T$
19:        Add offspring to $P'$
20:        Update $\theta^*$ if improved
21:    end for
22:    $P \leftarrow P'$
23: end while
24: return best weights $\theta^*$
```

## 7 Discussion

The proposed framework learns heuristics rather than explicit solutions. The neural network is optimized exclusively through downstream search performance, enabling learning in non-differentiable settings. This tightly coupled integration of evolutionary learning and Beam Search allows the system to adapt its guidance strategy to the structure of the problem instances.