Enhanced TPE Configuration Tuning Tool - User Manual

Project Overview

This project implements an enhanced Tree-structured Parzen Estimator (TPE) method for automated software system configuration tuning. The tool significantly outperforms raom search by incorporating system-specific optimization strategies, including adaptive transformation functions, noise handling, and simulated annealing refinement.

Features

Bayesian optimization using enhanced TPE for efficient exploration of configuration spaces.

System-specific transformation functions for optimal performance modeling.

Adaptive noise handling with calibrated parameters for different systems.

Two-phase simulated annealing refinement for local optimization.

Statistical analysis with Mann-Whitney U tests for result validation.

Comprehensive visualization of convergence curves and performance distributions.

Installation

1. Clone the Repository

```
git clone https://github.com/username/enhanced-tpe-tuning.git
cd enhanced-tpe-tuning
```

2. Install Dependencies

```
pip install -r requirements.txt
```

Dataset Description

The datasets folder should contain CSV files representing different configurable systems:

Columns (1 to n-1): Configuration parameters (discrete values)

Column n: Performance metric (a numeric value)

Supported Systems

System	Parameters	Optimization Type	Performance Metric
PostgreSQL	9	Minimization	Query execution time
spear	14	Minimization	Solving time
storm	12	Minimization	Processing latency
7z	8	Minimization	Compression performance

System	Parameters	Optimization Type	Performance Metric
Apache	8	Minimization	Response time
brotli	2	Minimization	Compression efficiency
LLVM	16	Minimization	Compilation performance
x264	10	Minimization	Encoding efficiency

Usage

1. Run TPE Optimization

Execute the main script to perform enhanced TPE optimization on all datasets:

```
python main.py
```

For specific datasets only:

```
python main.py --datasets PostgreSQL,storm,spear
```

Set custom parameters:

```
python main.py --budget 300 --num_runs 20 --repeats 5 --noise_scale 1e-4
```

2. Analyze Results

Generate statistical comparison between TPE and random search:

```
python stats_test.py
```

3. Visualize Results

Create convergence curves and performance distribution visualizations:

```
python visualize_tpe_search_results.py
```

Project Structure

```
enhanced-tpe-tuning/
├── datasets/                 # Input configuration datasets
├── tpe_results_improved/       # TPE optimization results
├── random_search/             # Baseline random search results
├── tpe_visualization_multi/     # Visualization outputs
├── main.py                   # Enhanced TPE implementation
├── main_random.py              # Random search baseline
├── stats_test.py              # Statistical comparison tools
├── visualize_tpe_search_results.py  # Results visualization
├── requirements.txt            # Dependencies
├── requirements.pdf             # Detailed dependency list
├── manual.pdf                 # This user guide
└── replication.pdf             # Instructions for reproducing results
```

Notes

The tool automatically detects system characteristics and applies appropriate transformation functions
For noise-sensitive systems (PostgreSQL, spear, storm), multiple measurements are taken with calibrated noise factors
TPE hyperparameters are optimized for each specific system
Simulated annealing refinement uses a two-phase strategy with adaptive search radius
Missing configurations are handled with system-specific penalty values
For detailed explanation of the methodology and technical implementation, please refer to the accompanying paper.