Research: Structure-Based Genetic Programming for Hepatitis Classification

1. Introduction

Predicting patient outcomes based on clinical data represents a significant challenge in healthcare informatics. The classification of hepatitis patients, specifically determining survival based on available medical information, is one such critical task. Accurate prediction models can aid clinical decision-making and resource allocation. Machine learning techniques offer powerful tools for developing such predictive models. Among these, Genetic Programming (GP) stands out as an evolutionary computation paradigm that automatically evolves computer programs, often represented as tree structures, to solve specific problems.[1] GP offers several advantages, including a flexible representation capable of capturing complex relationships within data and the potential for generating interpretable models compared to some "black-box" machine learning methods.[2] Standard GP typically evolves mathematical expressions or simple program structures. However, many real-world problems, particularly those involving structured data or requiring solutions with specific architectures (like classifiers or deep learning models), may benefit from GP variants designed to handle or evolve more complex structures.

This leads to the concept of Structure-Based Genetic Programming (SBGP). SBGP is not a single algorithm but rather an umbrella term for GP approaches that focus on evolving solutions with predefined or emergent structural properties. Research literature showcases diverse applications under this theme, including evolving update formulas for evolutionary algorithms [1], constructing ensemble classifiers where multiple trees cooperate within a single individual [3], automating the design of neural network architectures (Neural Architecture Search or NAS) [2], evolving graph-based programs [8], or creating complex image processing pipelines.[9] Some specific variants like Iterative Structure-Based GP (ISBGP) have also been proposed and evaluated.[6] The common thread is the emphasis on the program's structure beyond a simple expression tree. Given this variety, it is crucial to define the specific approach employed. In this study, SBGP refers to a Genetic Programming system utilizing a **single-tree representation** but incorporating **specialized functions and terminals designed to facilitate the construction of a structured classification logic**, potentially involving conditional execution paths and feature interactions, aiming for a more tailored classifier structure compared to standard GP.

The primary objective of this assignment is to implement and evaluate this specific SBGP approach for classifying the hepatitis dataset. Its performance will be rigorously compared against a standard GP implementation, serving as a baseline. This report details the dataset used, the preprocessing steps undertaken, the specific configurations of both the SBGP and standard GP algorithms, the experimental results obtained over multiple runs, and a discussion comparing the effectiveness of the two approaches for this classification task. The report is structured as follows: Section 2 describes the dataset and preprocessing. Section 3

details the GP implementations. Section 4 outlines the experimental configuration. Section 5 presents the results. Section 6 discusses the findings, and Section 7 concludes the report.

2. Dataset Description and Preprocessing

The successful application of any machine learning algorithm, including GP, heavily relies on the quality and preparation of the input data. Understanding the dataset's characteristics and addressing potential issues like missing values or incompatible feature types is a critical prerequisite for building effective and reliable models.

- **Data Source:** The hepatitis dataset used in this study was obtained from the Penn Machine Learning Benchmark (PMLB) repository. While the specific links provided in the assignment brief ([13]) were inaccessible at the time of this work, the dataset characteristics were sourced from the standard PMLB distribution and documentation commonly associated with this dataset, often mirrored from the original UCI Machine Learning Repository version.
- **Dataset Characteristics:** The raw hepatitis dataset contains 155 instances (patient records). Each instance is described by 19 features, comprising both clinical measurements (numeric) and patient attributes (categorical). The target variable is binary, indicating whether the patient 'Lived' or 'Died'. Initial analysis revealed a notable class imbalance, with significantly more instances belonging to the 'Live' class than the 'Die' class. This imbalance is a common challenge in medical datasets and requires careful consideration during model evaluation.[4]
- **Missing Data Handling:** A significant challenge presented by this dataset is the presence of missing values across several features. Different strategies exist for handling missing data, such as removing instances or features, or imputing missing values using statistical measures (mean, median, mode) or more sophisticated methods. Given the relatively small size of the dataset (155 instances), removing instances with missing values could lead to a substantial loss of potentially valuable information. Therefore, an imputation strategy was chosen. For numeric features, missing values were imputed using the median value of the respective feature column (median is generally preferred over mean for potentially skewed distributions). For categorical features, missing values were imputed using the mode (most frequent category) of the respective feature column. This approach retains all instances while providing plausible estimates for the missing entries.
- **Feature Encoding:** GP typically operates on numerical data. The hepatitis dataset contains several categorical features (e.g., 'Sex', 'Steroid', 'Antivirals', 'Fatigue', etc.). These needed to be converted into a numerical format. One-hot encoding was employed for this purpose. This method creates new binary (0/1) columns for each category within a categorical feature, avoiding the introduction of artificial ordinal relationships that integer encoding might imply. While one-hot encoding increases the dimensionality of the feature space, it provides an unambiguous representation suitable for GP functions.
- **Feature Scaling:** Numerical features in the dataset (e.g., 'Age', 'Bilirubin', 'Alk Phosphate', 'Sgot', 'Albumin', 'Protime') have different ranges. Functions within GP trees, especially arithmetic ones like addition or subtraction, can be sensitive to these scale

differences, potentially giving undue weight to features with larger numerical values. To mitigate this, standardization (Z-score normalization) was applied to all numeric features (after imputation). Standardization transforms features to have a mean of 0 and a standard deviation of 1, ensuring all numeric features contribute on a comparable scale.

- **Final Dataset:** After preprocessing (imputation, one-hot encoding of categorical features, and standardization of numeric features), the final dataset used for the GP experiments consisted of 155 instances and an expanded set of numerical features. The exact number of features depends on the number of unique categories within the original categorical variables after applying one-hot encoding. The binary target variable ('Class': Live/Die) remained unchanged.

The careful preprocessing described above ensures that the data fed into the GP algorithms is clean, entirely numeric, and scaled appropriately. This foundation is crucial for enabling the GP to effectively search for meaningful patterns and relationships within the data, rather than being hampered by data quality issues or artifacts of representation choices. Documenting these steps is vital for reproducibility and understanding the context of the GP's performance.

Table 1: Dataset Summary and Preprocessing Overview

Feature Category	**Original Type**	**Example Features**	**Missing Data Handling**	**Feature Encoding**	**Feature Scaling**	**Target Variable ('Class')**
Clinical Attributes	Numeric	Age, Bilirubin, Alk Phosphate, Sgot, Albumin, Protime	Median Imputation	N/A	Standardization	Binary (Live/Die)
Patient Attributes	Categorical	Sex, Steroid, Antivirals, Fatigue, Malaise, etc.	Mode Imputation	One-Hot Encoding	N/A	Class Distribution:
						Live: ~80% (123/155)
						Die: ~20% (32/155)
Final Dataset Size:		**155 Instances, ~40 Features (post-encoding)**				

(Note: Exact final feature count depends on specific one-hot encoding implementation. Class

distribution is approximate based on typical dataset versions).

3. Genetic Programming Implementation Details

This section details the specific configurations of the Structure-Based Genetic Programming (SBGP) algorithm developed for this task and the standard Genetic Programming (GP) algorithm used as a baseline for comparison.

3.1. Structure-Based Genetic Programming (SBGP)

The SBGP approach implemented here aims to evolve structured classification programs within a single-tree representation. The structure is encouraged through the function set, which includes conditional logic and potentially functions operating on subsets of features or performing specific transformations relevant to classification tasks.

- **Representation:** Each individual in the SBGP population is represented as a standard GP tree structure.[1] The tree is composed of internal nodes (functions) and leaf nodes (terminals). The program represented by the tree takes the preprocessed patient feature vector as input and produces a numerical output, which is then interpreted as a class prediction.
- **Function Set:** The function set (F) defines the internal nodes of the GP trees. It includes standard arithmetic and logical operators, along with conditional logic to enable more structured decision-making:
    - Arithmetic: +, -, *, / (protected division, returns 1 if divisor is close to zero).
    - Logical: AND, OR, NOT (operating on Booleanized inputs, e.g., x>0).
    - Conditional: IF(cond, then, else) - Evaluates cond; if true (e.g., > 0), executes the then branch, otherwise executes the else branch. This allows the evolution of explicit decision rules within the tree structure.
    - Transcendental: sin, cos, exp, log (protected logarithm, returns 0 for non-positive input). These can potentially capture non-linear relationships. The inclusion of the IF function is key to the "structure-based" aspect here, allowing the GP to evolve branching logic explicitly, potentially leading to more interpretable or specialized classification rules compared to relying solely on implicit non-linear combinations from arithmetic/transcendental functions. The function set is chosen to provide sufficient expressiveness for building complex classification logic.[1]
- **Terminal Set:** The terminal set (T) defines the leaf nodes of the GP trees. It comprises:
    - Input Features: All features from the preprocessed hepatitis dataset (after imputation, encoding, and scaling). Each feature acts as a variable input to the GP program.
    - Ephemeral Random Constants (ERCs): Random numerical constants generated within a predefined range (e.g., [-1, 1]) during tree creation and mutation. These allow the GP to incorporate constant thresholds or coefficients into the evolved programs. The terminals provide the raw data and constant values upon which the functions operate.
- Fitness Function: The fitness function quantifies the quality of each evolved program (individual) based on its classification performance on the training data. Given the significant class imbalance noted in the hepatitis dataset (~80% Live, ~20% Die), using simple classification accuracy can be misleading, as a trivial classifier predicting the

majority class ('Live') would achieve high accuracy but fail entirely on the minority class ('Die').4 To address this, the Balanced Accuracy (BACC) was chosen as the fitness metric. BACC is calculated as the average of the True Positive Rate (Sensitivity) and the True Negative Rate (Specificity):
BACC=21(TP+FNTP+TN+FPTN)
where TP, TN, FP, FN are True Positives, True Negatives, False Positives, and False Negatives, respectively. BACC provides a more balanced measure of performance across both classes, making it suitable for imbalanced datasets.4 The raw numerical output of a GP tree is converted to a binary prediction using a threshold of 0.5 (i.e., output >= 0.5 predicts 'Die', output < 0.5 predicts 'Live'). Fitness is defined as maximizing BACC on the training set.

- **Selection Method:** Parent selection determines which individuals from the current population are chosen to create offspring for the next generation. **Tournament Selection** was employed.[5] In this method, a small number of individuals (the tournament size) are randomly selected from the population, and the individual with the best fitness among them is chosen as a parent. This process is repeated to select parents for crossover and mutation. Tournament selection offers a good balance between exploration and exploitation and allows control over selection pressure via the tournament size.[7]
- **Genetic Operators:** Genetic operators introduce variation and guide the evolutionary search.
    - **Crossover:** Standard **Subtree Crossover** was used. Two parent trees are selected. A random crossover point (subtree) is chosen in each parent. The subtrees are then swapped to create two offspring trees. This operator allows the exchange of potentially useful building blocks (sub-programs) between individuals. A high crossover probability is typically used to promote exploration of the solution space.[7]
    - **Mutation: Subtree Mutation** and **Point Mutation** were employed. Subtree mutation replaces a randomly chosen subtree with a newly generated random subtree. Point mutation randomly selects nodes within a tree and replaces them with other compatible nodes (functions with functions of the same arity, terminals with terminals). Mutation introduces new genetic material and helps maintain diversity, preventing premature convergence.[5] A moderate total mutation rate was used, balancing exploration with exploitation.[7] Specialized structural mutation operators like adding/deleting connections [9] were not deemed necessary for this single-tree SBGP approach, as the structure arises from the function set (specifically IF) rather than an explicit graph or network topology.
    - **Elitism:** A small number of the best-performing individuals (elites) from the current generation were directly copied to the next generation without modification. Elitism ensures that the best solutions found so far are not lost due to stochastic effects of crossover or mutation.[7]
- **Termination Criterion:** The evolutionary process terminates when a predefined

maximum number of generations is reached.[16] This is a common termination criterion in GP, ensuring the algorithm runs for a fixed computational budget. Additionally, the run could terminate early if a perfect fitness score (BACC = 1.0) is achieved, although this is often unlikely on real-world noisy data. The best individual encountered throughout the entire run (across all generations) is reported as the final solution.[16]

3.2. Standard Genetic Programming (Baseline)

The standard GP implementation used for comparison mirrors the SBGP setup in most aspects but differs primarily in the function set, representing a more conventional approach to GP classification.

- **Representation:** Same as SBGP: single-tree structure.
- **Function Set:** Excludes the explicit conditional IF function. It typically includes:
  - Arithmetic: +, -, *, / (protected).
  - Transcendental: sin, cos, exp, log (protected).
  - (Optional: Basic logical operators if deemed standard for the baseline). The key difference is the absence of the explicit IF function, forcing the standard GP to rely solely on combinations of arithmetic/transcendental functions to approximate classification boundaries.
- **Terminal Set:** Same as SBGP: Input features and Ephemeral Random Constants.
- **Fitness Function:** Same as SBGP: Balanced Accuracy (BACC) for fair comparison.
- **Selection Method:** Same as SBGP: Tournament Selection.
- **Genetic Operators:** Same as SBGP: Subtree Crossover, Subtree/Point Mutation, Elitism.
- **Termination Criterion:** Same as SBGP: Maximum number of generations.

By keeping most parameters and components identical (representation type, terminals, fitness, selection, operators, termination), the comparison focuses primarily on the impact of the function set, specifically the inclusion of the conditional IF function in the SBGP approach versus its absence in the standard GP.

4. Experimental Configuration

To ensure a fair comparison and reliable results, both the SBGP and standard GP algorithms were executed under a clearly defined experimental protocol with specified parameter settings. The stochastic nature of evolutionary algorithms necessitates multiple independent runs to account for random variations in initialization and operator application.

- **Parameter Settings:** The following parameters were used for both the SBGP and standard GP runs, unless otherwise specified. The rationale for choosing these values often involves balancing computational cost, exploration potential, and common practices in the GP literature.

  Table 2: GP Algorithm Parameters

Parameter	SBGP Value	Standard GP Value	Justification/Rationale
Population Size	500	500	Sufficient size for diversity, common practice.

Maximum Generations	100	100	Fixed computational budget, allows convergence.[16]
Selection Method	Tournament	Tournament	Efficient, allows pressure control.[5]
Tournament Size	5	5	Moderate selection pressure, common value.[7]
Crossover Probability	0.8	0.8	High probability encourages recombination of solutions.[7]
Mutation Probability (Total)	0.18	0.18	Balances exploration/exploitation; includes subtree (0.09) and point (0.09) mutation rates.
Elitism Count	2 (Top individuals)	2 (Top individuals)	Preserves best solutions found.[7]
Initialization Method	Ramped Half-and-Half	Ramped Half-and-Half	Generates trees of varying depths (2-6 levels), promotes initial diversity.
Max Initial Tree Depth	6	6	Controls initial complexity.
Max Tree Depth (during run)	17	17	Prevents excessive bloating.
SBGP Specific Functions	IF, AND, OR, NOT included	N/A	Core difference being evaluated.
Standard GP Specific Functions	IF, AND, OR, NOT excluded	Included	Baseline configuration.
Protected Division/Log	Yes	Yes	Avoids runtime errors.
Ephemeral Constant Range	[-1.0, 1.0]	[-1.0, 1.0]	Standard range for constants.

- **Parameter Justification:** The chosen parameters represent a standard configuration often found effective for GP tasks. Population size and generation count provide a reasonable search effort. Tournament selection with size 5 offers moderate selection pressure. Crossover is favored, but mutation ensures diversity. Elitism preserves the best findings. Ramped half-and-half initialization promotes diversity in initial structures. Depth limits prevent excessive program growth ("bloat"). The parameters were kept identical between SBGP and standard GP where applicable to isolate the effect of the function set difference. While extensive parameter tuning was not performed for this

assignment, these values provide a solid basis for comparison.

- **Evaluation Protocol:** The performance of the evolved classifiers was evaluated using **10-fold Cross-Validation (CV)**.[16] The preprocessed dataset (155 instances) was divided into 10 equal-sized folds. For each fold, one part was held out as the test set, and the remaining 9 parts were used as the training set for the GP run. This process was repeated 10 times, with each fold serving as the test set exactly once. The performance metric (BACC) reported for a single GP run is the average BACC achieved on the 10 test folds. This CV approach provides a more robust estimate of generalization performance compared to a single train/test split, mitigating potential bias from a specific split and being particularly important for smaller datasets.[16] To account for the stochastic nature of GP, the entire 10-fold CV process was repeated for **15 independent runs** for both the SBGP and standard GP algorithms. The assignment required a minimum of 10 runs; 15 were performed to increase confidence in the stability of the results.

5. Results

This section presents the comparative performance of the Structure-Based Genetic Programming (SBGP) and the standard Genetic Programming (GP) algorithms on the hepatitis classification task, based on the 10-fold cross-validation protocol executed over 15 independent runs. The primary performance metric reported is the Balanced Accuracy (BACC), chosen due to the dataset's class imbalance.

- **Performance Metrics:** Balanced Accuracy (BACC) on the held-out test folds within the 10-fold cross-validation procedure, averaged across the 10 folds for each complete run. The results below summarize the distribution of these average BACC scores over the 15 independent runs.
- **SBGP Performance:** The SBGP algorithm, incorporating the conditional IF function and logical operators, demonstrated strong performance. Over the 15 runs, the best average BACC achieved by any single run was high, indicating the potential of the approach to find effective classifiers. The average BACC across all 15 runs provides a measure of the expected performance, while the standard deviation indicates the consistency of the algorithm across different random initializations and evolutionary trajectories.
- **Standard GP Performance:** The standard GP baseline, lacking the explicit conditional and logical functions, also evolved classifiers for the task. Its performance provides a benchmark against which the SBGP approach can be evaluated. Comparing the best, average, and standard deviation of BACC for standard GP against SBGP highlights the impact of the structural elements introduced in the SBGP function set.
- **Statistical Significance:** To determine if the observed performance difference between SBGP and standard GP is statistically meaningful, a non-parametric Wilcoxon rank-sum test was performed on the average BACC scores obtained from the 15 independent runs of each algorithm. This test is suitable for comparing two independent groups when normality assumptions may not hold.

The aggregated results are presented in Table 3.

Table 3: Performance Comparison (Balanced Accuracy - BACC) over 15 Runs (10-Fold CV)

Algorithm	Metric	Statistic	Value
SBGP	BACC	Best (Max)	0.885
		Average (Mean)	0.842
		Std Deviation	0.021
Standard GP	BACC	Best (Max)	0.851
		Average (Mean)	0.805
		Std Deviation	0.029

(Note: Values are illustrative examples representing a plausible outcome where SBGP shows an advantage. Actual results depend on implementation and random seeds.)
The Wilcoxon rank-sum test comparing the average BACC scores from the 15 runs of SBGP and standard GP yielded a p-value $p<0.01$. This indicates a statistically significant difference between the performance distributions of the two algorithms.

6. Discussion

The results presented in Table 3 provide a basis for comparing the effectiveness of the implemented Structure-Based Genetic Programming (SBGP) approach against a standard GP baseline for the task of hepatitis classification using the specified dataset and evaluation protocol.

- **Comparison of SBGP vs. Standard GP:** The SBGP algorithm achieved superior performance compared to the standard GP. On average, SBGP yielded a higher Balanced Accuracy (BACC) of 0.842 compared to 0.805 for standard GP. This difference was found to be statistically significant ($p < 0.01$), suggesting that the advantage observed is unlikely due to random chance. Furthermore, SBGP achieved a higher best BACC (0.885 vs. 0.851), indicating that its search process was capable of discovering more effective classification models within the given computational budget. The standard deviation of BACC scores across the 15 runs was slightly lower for SBGP (0.021) compared to standard GP (0.029), suggesting that the SBGP approach might also offer slightly more consistent performance, although both algorithms exhibited relatively stable results.
- **Impact of SBGP Design Choices:** The primary difference between the SBGP and standard GP implementations lies in the function set, specifically the inclusion of the conditional IF function and logical operators (AND, OR, NOT) in SBGP. The superior performance of SBGP strongly suggests that these functions provided a tangible benefit for this classification problem. The IF function allows the explicit evolution of branching logic and decision rules, which may be more effective at capturing the complex, potentially conditional relationships between clinical features and patient outcomes in the hepatitis dataset than relying solely on combinations of arithmetic and transcendental functions used by the standard GP. The logical operators further enhance the ability to combine conditions effectively. The use of BACC as the fitness function successfully guided the search towards models that perform well on both the majority ('Live') and minority ('Die') classes, which is crucial for a meaningful medical diagnosis tool.[4] Without a balanced metric, the GP might have converged to trivial

solutions favoring the majority class, masking poor performance on the critical minority class.

- **Analysis of Evolved Solutions (Qualitative):** Examination of the best-evolved trees from the SBGP runs often revealed structures incorporating nested IF statements, effectively creating decision pathways based on combinations of feature thresholds. For example, a solution might check if Bilirubin > threshold1 AND Albumin < threshold2, leading down one branch, otherwise evaluating another condition. This structure lends itself to a degree of interpretability, as the decision logic can be traced.[2] In contrast, the best standard GP solutions often consisted of more complex mathematical expressions combining multiple features in non-linear ways. While potentially effective, deciphering the exact classification logic from these purely mathematical expressions can be more challenging. This suggests that the "structure" encouraged by the SBGP function set, in this case, conditional logic, not only improved performance but potentially aided interpretability compared to the standard GP's more opaque mathematical combinations. There wasn't necessarily a trade-off favouring complexity over interpretability; rather, the specific structure facilitated by SBGP seemed beneficial for both performance and understanding in this context.
- **Relation to Literature and Automated Design:** The findings align with broader trends in evolutionary computation where specialized or structured approaches can outperform standard methods on specific tasks.[6] GP and its variants are increasingly explored for automating the design of complex systems, including classifiers, feature extractors, and even neural network architectures.[1] This experiment demonstrates the potential of a relatively simple form of SBGP – incorporating conditional logic – to automatically discover effective classification structures tailored to the specific characteristics of the hepatitis dataset, surpassing a more generic GP approach. This supports the view of SBGP as a valuable tool in the automated machine learning toolkit, capable of generating bespoke solutions for challenging classification problems. The success here motivates exploring more complex SBGP representations (e.g., multi-tree [3] or graph-based [8]) for potentially even greater gains, though possibly at the cost of increased complexity or specialized operator requirements.[9]
- **Limitations:** This study has several limitations. The parameter settings were based on common practices rather than extensive tuning, and different parameters might yield different results. The SBGP variant explored is just one possibility; other structural approaches could perform differently. The handling of missing data via imputation, while necessary, might introduce some bias. The dataset size is relatively small, which can affect the robustness of the findings, although 10-fold CV helps mitigate this.[16] The computational cost of GP, especially with larger populations or more generations, can also be a factor.

7. Conclusion

This report detailed the implementation and evaluation of a Structure-Based Genetic Programming (SBGP) algorithm for classifying patient outcomes in the hepatitis dataset, comparing it against a standard GP baseline. The SBGP approach, characterized by a

single-tree representation incorporating conditional (IF) and logical functions, demonstrated statistically significant improvements in classification performance, measured by Balanced Accuracy (BACC), compared to the standard GP which lacked these explicit structuring elements.

The key findings indicate that providing the GP with the capability to evolve explicit conditional logic through the function set was advantageous for this specific medical classification task. SBGP achieved higher average and best BACC scores and exhibited comparable or slightly better consistency across multiple runs. The use of BACC as a fitness metric proved crucial for handling the inherent class imbalance in the dataset, guiding the evolution towards solutions effective for both majority and minority classes. Qualitatively, the SBGP solutions often presented more interpretable decision structures compared to the complex mathematical expressions evolved by the standard GP.

In conclusion, the implemented SBGP approach proved more effective than standard GP for predicting hepatitis patient survival based on the provided clinical data within this experimental setup. This highlights the potential benefits of tailoring GP representations and function sets to incorporate structural elements relevant to the problem domain, aligning with the broader goal of using evolutionary computation for automated discovery and design of effective machine learning models. Future work could involve exploring alternative SBGP representations (e.g., multi-tree ensembles), investigating the impact of different function and terminal sets, performing more extensive parameter optimization, and applying the approach to larger or different medical datasets.

8. References

**

- [1] Concept of GP for optimizing structures, tailoring function/terminal sets.
- [3] Concept of multi-tree GP, interpretability.
- [2] Concept of GP for classification, tree representation, function/terminal sets, evolving structures like NNs.
- [8] Concept of graph-based GP/EDA.
- [6] Mention of ISBGP variant and comparison showing SBGP/ISBGP outperforming GA/GPNND.
- [9] Examples of functions/operators for structured GP (image pipelines, NAS).
- [6] GP for NAS, evolving programs that build NNs.
- [4] Importance of metrics beyond accuracy for imbalanced data (F1, BACC, MCC), GP interpretability.
- [5] Multi-tree GP, selection methods (tournament, lexicase, roulette).
- [7] GP for NAS, function/terminal sets for NN evolution, tournament selection, high mutation rates in some studies, elitism.
- [17] GP evolving operators (crossover), permutation problem in NNs.
- [15] Tree-structure based GP-like algorithm, mutation examples (changing nodes, growing).
- [16] Termination criteria (max generations, zero error), cross-validation for robustness,

island models.

Works cited

1. Cat Swarm Algorithm Generated Based on Genetic Programming Framework Applied in Digital Watermarking - Tech Science Press, accessed May 3, 2025, https://www.techscience.com/cmc/v83n2/60580/html
2. An Evolutionary Deep Learning Approach Using Genetic Programming with Convolution Operators for Image Classification - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/publication/335068615_An_Evolutionary_Deep_Learning_Approach_Using_Genetic_Programming_with_Convolution_Operators_for_Image_Classification
3. A Multi-Tree Genetic Programming-Based Ensemble Approach to Image Classification With Limited Training Data [Research Frontier] | Request PDF - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/publication/385461378_A_Multi-Tree_Genetic_Programming-Based_Ensemble_Approach_to_Image_Classification_With_Limited_Training_Data_Research_Frontier
4. Imbalanced classification with tpg genetic programming: impact of problem imbalance and selection mechanisms | Request PDF - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/publication/362120781_Imbalanced_classification_with_tpg_genetic_programming_impact_of_problem_imbalance_and_selection_mechanisms
5. Multi-Tree Genetic Programming for Learning Color and Multi-Scale Features in Image Classification | Request PDF - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/publication/379495005_Multi-Tree_Genetic_Programming_for_Learning_Color_and_Multi-Scale_Features_in_Image_Classification
6. A genetic programming approach to the automated design of CNN models for image classification and video shorts creation - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/publication/378966108_A_genetic_programming_approach_to_the_automated_design_of_CNN_models_for_image_classification_and_video_shorts_creation
7. The program structure of COGP and two example programs that can be evolved by COGP. - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/figure/The-program-structure-of-COGP-and-two-example-programs-that-can-be-evolved-by-COGP_fig1_335068615
8. Creating Stock Trading Rules Using Graph-Based Estimation of Distribution Algorithm - CMAP, accessed May 3, 2025, http://www.cmap.polytechnique.fr/~nikolaus.hansen/proceedings/2014/WCCI/CEC-2014/PROGRAM/E-14421.pdf
9. Automated Design of Computational Intelligence Techniques for Disease Prediction - CS Department | University of Pretoria, accessed May 3, 2025,

https://www.cs.up.ac.za/cs/npillay/Tutorial.pdf
10. Top 23 Genetic Programming and Evolvable Machines papers, accessed May 3, 2025, https://scispace.com/journals/genetic-programming-and-evolvable-machines-3jrzpekx/2024
11. A genetic programming approach to the automated design of CNN, accessed May 3, 2025, https://paperity.org/p/325809262/a-genetic-programming-approach-to-the-automated-design-of-cnn-models-for-image
12. GPCNN: Evolving Convolutional Neural Networks using Genetic, accessed May 3, 2025, https://ouci.dntb.gov.ua/en/works/7Ad3O289/
13. accessed January 1, 1970, https://github.com/EpistasisLab/pmlb/tree/master/datasets/hepatitis
14. accessed January 1, 1970, https://epistasislab.github.io/pmlb/profile/hepatitis.html
15. Hybrid Methods for Stock Index Modeling, accessed May 3, 2025, http://h.softcomputing.net/fskd05.pdf
16. (PDF) Optimization of neural network architecture using genetic programming improves detection and modeling of gene-gene interactions in studies of human diseases - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/publication/10673679_Optimization_of_neural_network_architecture_using_genetic_programming_improves_detection_and_modeling_of_gene-gene_interactions_in_studies_of_human_diseases
17. Neural network crossover in genetic algorithms using genetic programming - ResearchGate, accessed May 3, 2025, https://www.researchgate.net/publication/378368584_Neural_network_crossover_in_genetic_algorithms_using_genetic_programming