Phase-alternation and tree-initialization to facilitate interpretable rule-based machine learning

Gabriel Lipschutz-Villa, Harsh Bandhey, Khoi Dinh, Michael Heider, Malek Kamoun, Ryan Urbanowicz

Angaben zur Veröffentlichung / Publication details:

Lipschutz-Villa, Gabriel, Harsh Bandhey, Khoi Dinh, Michael Heider, Malek Kamoun, and Ryan Urbanowicz. 2026. "Phase-alternation and tree-initialization to facilitate interpretable rule-based machine learning." In *GECCO '26: proceedings of the Genetic and Evolutionary Computation Conference, Centro Internacional de Convenciones CIC-ANDE, San Jose, Costa Rica, July 13-17, 2026*, edited by Leonardo Trujillo and Ting Hu, 300–309. New York, NY: ACM. https://doi.org/10.1145/3795095.3805127.

Phase-Alternation and Tree-Initialization to Facilitate Interpretable Rule-based Machine Learning

Gabriel Lipschutz-Villa
gabelip@sas.upenn.edu
University of Pennsylvania
Philadelphia, PA, USA

Harsh Bandhey
harsh.bandhey@cshs.org
Cedars Sinai Health Sciences University
Los Angeles, CA, USA

Khoi Dinh
khoidinh@seas.upenn.edu
University of Pennsylvania
Philadelphia, PA, USA

Michael Heider
michael.heider@uni-a.de
Universität Augsburg
Augsburg, Germany

Malek Kamoun
malekkam@pennmedicine.upenn.edu
University of Pennsylvania
Philadelphia, PA, USA

Ryan J. Urbanowicz*†
ryan.urbanowicz@cshs.org
Cedars Sinai Health Sciences University
Los Angeles, CA, USA

Abstract

Interpretable machine learning is important for scientific, biomedical, and other high-stakes domains where transparency, trust, and human understanding of models are required. Recently, an evolutionary rule-based machine learning algorithm called HEROS was proposed, which separated rule from rule-set discovery each with distinct multi-objective optimization strategies yielding a two-phase, noise agnostic framework that found accurate, highly compact, and interpretable solutions on a diverse set of challenging benchmarks without requiring hyperparameter optimizations. However, HEROS is currently limited by its strictly sequential two-phase design which requires committing in advance to a fixed number of rule-discovery iterations. This work extends HEROS with a phase alternation scheme interleaving rule discovery and rule-set optimization. This enables refinement of both tasks throughout training in an 'anytime-algorithm' manner. We also introduce a tree-based rule initialization strategy to accelerate early-stage rule discovery. These extensions aim to preserve the empirical performance of the original HEROS framework under a similar computational budget while providing greater flexibility for algorithm application and problem scalability. Using the same diverse set of benchmark datasets, we evaluate and compare these HEROS extensions to the original HEROS algorithm and established rule-based learning systems including RIPPER and BioHEL. We demonstrate the advantages of this extended HEROS framework.

CCS Concepts

• **Computing methodologies** → **Rule learning**; **Supervised learning**; **Genetic algorithms**; *Instance-based learning*; *Batch learning*; Cross-validation; Model development and analysis.

Keywords

Rule-based Machine Learning, Learning Classifier Systems, Supervised Learning, Interpretability, Multi-objective Optimization

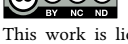
ACM Reference Format:
Gabriel Lipschutz-Villa, Harsh Bandhey, Khoi Dinh, Michael Heider, Malek Kamoun, and Ryan J. Urbanowicz. 2026. Phase-Alternation and Tree-Initialization to Facilitate Interpretable Rule-based Machine Learning. In *Genetic and Evolutionary Computation Conference (GECCO '26), July 13–17, 2026, San Jose, Costa Rica.*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3795095.3805127

1 Introduction

Interpretability is often essential for machine learning (ML) utilized in scientific, biomedical, and other high-stakes fields, where model transparency, trust, and the capacity to understand model reasoning are as important as predictive performance [32, 34]. While modern, so-called, 'black-box' models can yield high accuracy, their hidden internal representations lead to limited utility in situations where there is a need for causal knowledge, validation by domain experts, or regulatory supervision [34]. Additionally, relying on 'model explanation' or 'feature importance estimation' strategies, such as Shapley values, LIME, or permutation feature importance for 'black-box' model interpretation, provide only a limited, estimated, and potentially biased understanding of model decision making [25, 27, 46]. Differently, highly interpretable modeling strategies, such as decision trees, generally achieve lesser predictive performance, in particular on more complex modeling tasks [45]. Consequently, there has been persistent interest in rule-based ML (RBML) strategies as principled methodologies for developing interpretable models that can also tackle complexity [1, 18, 42].

RBML algorithms encode knowledge as a set of symbolic rules of form: **IF** *condition(s)* **THEN** *outcome*s, which partition decision boundaries into localized, comprehensible elements [18]. This paradigm encompasses greedy rule learners like RIPPER [11], evolutionary rule-based systems such as Learning Classifier Systems (LCSs) [37, 47], and methodologies that integrate evolutionary search with greedy or heuristic techniques [1, 15]. Such methods are capable of finding patterns that are nonlinear, diverse, complex, and noisy, while still being easy to interpret; a task often not possible with modern ML approaches like deep learning [18, 32].

*Shared First-Authorship
†Corresponding Author



GECCO '26, San Jose, Costa Rica

ACM ISBN 979-8-4007-2487-9/2026/07
https://doi.org/10.1145/3795095.3805127

In evolutionary RBML, a persistent difficulty is the simultaneous optimization of individual rules vs. the compact set of rules comprising the trained model. Michigan-style (MIC-style) systems evolve a *population* {*P*} of rules that are individually optimized and where all of {*P*} serves as the model (generally after some rule-compaction). Differently, Pittsburgh-style (PIT-style) systems evolve a *model population* {*MP*} of rule-sets similar to traditional genetic algorithms (GAs) [6]. Each has its own benefits in terms of scalability, compactness, and adaptability [8, 37]. Proposed hybrid methods have endeavored to integrate these paradigms to achieve both optimal performance and model compactness/interpretability, e.g. BioHEL [1, 16], fuzzy LCSs [26, 35], and SupRB [20, 23]. In particular, the fuzzy LCSs and SupRB introduced systems that separated rule from rule-set optimization. However, the designs of these methods make limiting assumptions, such as; (A) dependence on a single fitness function covering both rule and rule-set optimization [1], (B) use of a single-objective fitness function (e.g. accuracy) [35], (C) a preset weighting of the multiple objectives [1, 21, 26], (D) reliance on a greedy approach to rule-set formation [1, 35], and/or (E) the imprecision of fuzzification [26, 35]. When faced with a diversity of problems characterized by differing noise levels and/or underlying complexity, these assumptions can make them less robust [32].

Recently, the Heuristic Evolutionary Rule Optimization System (HEROS) was developed as a two-phase, hybrid, 'noise-agnostic' RBML framework that also separates rule vs. rule-set discovery and optimization, but introduced distinct, non-weighted, multi-objective optimization strategies for each task [29]. Phase I employed Pareto-inspired rule optimization in {*P*} maximizing *rule accuracy* and/or *instance coverage* to find a variety of high-quality candidate rules suitable to the possibility of either 'noisy' or 'clean' associations. Phase II employed NSGA-II-style [13] optimization in {*MP*} maximizing rule-set balanced accuracy and minimizing rule-set size. This strategy produced precise, compact, and fundamentally interpretable models for clean vs. noisy and simple vs. complex benchmark problems without having to select objective weights or dataset-specific hyperparameter settings.

While promising, HEROS was implemented as a 'staged' algorithm, operating in a strictly sequential manner, i.e. Phase I followed by Phase II. This assumption is problematic for real-world applications where the number of iterations required to reliably discover all 'ideal' or 'near-ideal' rules is unknown *apriori*. Ending Phase I too soon leaves Phase 2 with a suboptimal set of rules to combine into rule-sets, while running Phase I for too long wastes computation. This staged architecture disqualifies HEROS as an '*anytime algorithm*', which can describe most evolutionary algorithms. This makes it difficult to do things like; continue running the algorithm for additional training iterations rather than restarting from scratch, or stopping early if a satisfactory solution was found [12].

Previously, Fuzzy LCSs [26, 35] and SupRB [22, 23] explored *phase alternation* to interleave the progression of rule vs. and rule-set optimization with a fixed or heuristic schedule. Alternation preserves the '*anytime algorithm*' status of the two-phase approach and allows candidate rule-sets to affect ongoing rule discovery. These algorithms showed that alternation was possible, but relied on field-specific assumptions, and were only tested on small or 'well-behaved' datasets. Importantly, naive alternation strategies can lead to inefficient learning: when alternation starts too soon, iterations of rule-set optimization can be wasted with only suboptimal rules available in {*P*} [19]. This observation underscores a significant tension: although phase alternation enhances flexibility, facilitates continuous learning, and provides opportunities for *mutualistic co-evolution* between rules and rule-sets, it can waste rule-set optimization iterations and slow down overall algorithm convergence if rule discovery does not advance rapidly enough.

A promising way to lower this cost is to speed up the discovery of early rules, making sure that even the first cycles of alternation work with a meaningful collection of candidate rules. Previous research in RBML and genetic rule learning has demonstrated that informed *rule initialization*, especially through the utilization of rules derived from *decision trees*, can significantly enhance convergence speed and solution quality by directing the initial population towards informative areas of the search space [10, 28, 36]. Initialization based on decision trees has been investigated in both evolutionary rule learners and hybrid symbolic systems, where tree paths inherently correspond to concise, human-readable rules [9, 10].

The goal of the present study is to extend HEROS [29] to overcome the practical constraints of staged modeling while maintaining or improving its ability to accurately, compactly, and agnostically tackle a diversity of problem types without making objective weighting or hyperparameter assumptions. Specifically, we introduce a tree-based rule initialization strategy that populates Phase I with high-quality candidate rules derived from decision-trees [10, 12]. We then combine this with a simple phase alternation scheme that interleaves rule vs. rule-set optimization allowing HEROS to operate as an 'anytime algorithm'. This will make HEROS a significantly more flexible, extendable, and practical algorithm for a variety of real-world predictive modeling applications.

We experimentally assess our enhanced HEROS framework first with tree-initialization only, then adding phase alternation, across the same diversity of clean vs. noisy, simple vs. complex benchmark datasets used to compare original HEROS to the RBML, 'ExSTraCS' [29]. This study omits direct comparison to ExSTraCS, since HEROS largely demonstrated the same or superior predictive performance as well as dramatically more compact and interpretable rule-sets [29]. For the first time, this study also includes a direct comparison of 'original' and 'extended' HEROS to other established, RBML algorithms including RIPPER and BioHEL.

2 Methods

We provide a summary of the original HEROS algorithm along with details of our proposed *phase alternation* and *tree-based rule initialization* strategies. Next we briefly describe the RIPPER and BioHEL algorithms, and then detail our experimental evaluation across all compared algorithm configurations.

2.1 The HEROS Algorithm

We review the key steps, algorithmic elements, and components of the original HEROS algorithm and newly proposed extensions. While this description focuses on a higher-level understanding of HEROS, a more detailed explanation can be found in [29].

2.1.1 Overview. HEROS was descended from a lineage of MIC-style RBMLs including XCS [47], UCS [5], and ExSTraCS [42] and adopted the concept of separating rule vs. rule-set optimization

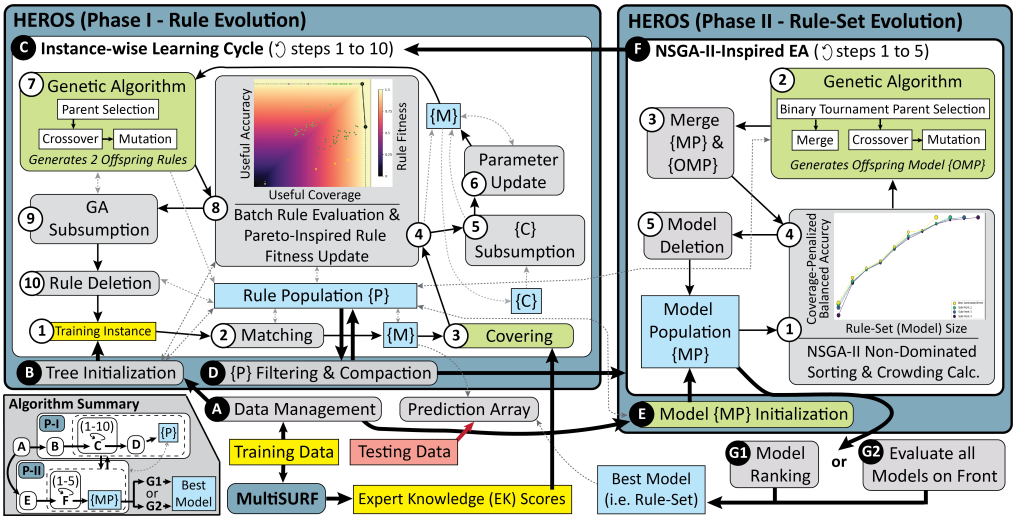


Figure 1: Schematic of Extended HEROS Algorithm. The bottom-left box summarizes algorithm phases, components (letters), and learning cycle steps (numbers). Yellow boxes indicate training data or EK weights, light-blue boxes indicate rule or model sets, green boxes indicate rule or model discovery elements, and gray boxes indicate other algorithm elements. Thick-black lines indicate algorithm flow between phases and components, thin black lines indicate flow between cycle steps, and dotted grey lines indicate other connections. Novel extensions include *tree-initialization* (see B) and *phase alternation* (see connections from D to F and F to C).

[23, 26, 35]. HEROS was designed for evolutionary ML modeling in structured, supervised data and allows mixed feature types, binary or multiclass outcomes and missing data. Figure 1 provides a schematic of algorithm elements and connections, which we reference throughout this section. Both *data management* and the *prediction array* are components shared by both phases.

HEROS hyperparameters are given in Table 1 along with default hyperparameter values used in this study. The first 4 hyperparameters (highlighted) can impact performance since they control allotted computational resources, however they do not assert assumptions about the nature of the underlying problem beyond 'resources needed to reach convergence'. The last three hyperparameters (highlighted) are the only ones that were varied to run the analyses presented in this paper. The hyperparamters (`rule_pop_init` and `alternate`) control the newly added extensions, and `random_state` is updated from 0-19 across 20 random seed runs. Some of these hyperparameters will be referenced below.

2.1.2 Phase I - Rule Optimization. Phase I of HEROS is a MIC-style LCS, inspired by scikit-ExSTraCS [49, 50]. Following Figure 1, Phase I begins with **(A)** *data management* which automatically calculates a rule specificity limit, and scales any expert knowledge (EK) weights to the range (0,1) [39, 42]. Next, **(B)** *tree-based rule initialization* can be utilized. This is the first extension we introduce to HEROS (detailed in section 2.3). Notably, tree initialization occurs only once with or without phase alternation. Differently, original HEROS would start with an empty rule population $\{P\}$.

Rules. Rules are represented as IF:THEN (i.e. condition:action) expressions using a mixed discrete-continuous attribute-list rule representation similar to ALKR [3, 38]. A rule's *condition* is defined as a set of specified features and their corresponding values. Unspecified features do not impact whether a rule *matches* a given instance, allowing for generalization. The rule's *action* is the class/outcome it predicts. Rules also maintain associated parameters such as rule *accuracy*, *fitness* and *numerosity* (detailed in [29]).

Table 1: HEROS Algorithm Hyperparameters.

Hyperparameter	Default	Description
`Rule_I`	200,000	Number of Phase I learning iterations
`Rule_P`	2,000	Maximum rule population size
`Model_I`	200	Number of Phase II learning iterations
`Model_P`	100	Maximum model population size
ν	1	Accuracy pressure (1 is neutral)
β	0.2	Average $\{M\}$ size learning rate
θ_{sel}	0.5	$\{M\}$ proportion for tournament sel.
`p_cross`	0.8	Probability of applying crossover
`p_mut`	0.04	Probability of applying mutation
`p_merge`	0.1	Probability of applying merge
`new_gen`	1.0	Controls size of $\{OMP\}$ as f(`Model_P`)
`model_pop_init`	*target_acc*	Model initialization method
`subsumption`	*both*	Use GA, $\{C\}$, or 'both' subsumptions
`rsl`	*auto*	Manually specify rule specificity limit
`compaction`	*sub*	Rule compaction method
`rule_pop_init`	*random* or *tree_init*	Rule initialization method
`alternate`	0 or 5	Number of alternations
`random_state`	(0,20)	Applied random seed

Instance-wise Learning Cycle. Next, **(C)** an instance-wise learning cycle is applied for (`Rule_I`) iterations. Each iteration, **(C−1)** the cycle starts by focusing on a single training instance, then **(C−2)** forms a match set $\{M\}$ of rules in $\{P\}$ that match specified features of the current instance. If $\{M\}$ is empty or no rules in $\{M\}$ have the correct action, **(C−3)** a *covering* operator is applied to generate a new rule who's action will later be assigned as the one that maximizes *'useful' accuracy* (UA_r) [43] of the rule. HEROS employs EK covering [39] using EK scores from the *MultiSURF* algorithm [44].

Next, **(C-4)** *batch evaluation* is applied to the new rule, using all training data to update its internal parameters, check whether the new rule is *non-dominated* on the current *rule Pareto-front*, and if so, update the fitness of all rules in $\{P\}$. HEROS employs a Pareto-inspired multi-objective rule fitness that seeks to maximize UA_r and/or *'useful' rule coverage* [43] as detailed in [29]. Next, **(C-5)** $\{C\}$ *subsumption* is applied (see [29]). Next, **(C-6)** a brief rule parameter update for rules in $\{M\}$ that later impacts deletion (see [29]). Next, **(C-7)** a GA is employed for rule discovery with crossover and mutation operators. EK scores are used to probabilistically guide mutation. Notably, the GA selects parent rules from $\{M\}$ since it's possible for $\{C\}$ to be empty (see [29]). Now, **(C-8)** each offspring rule is batch evaluated, has it's action assigned, is checked for Pareto non-domination, and has fitness updated (like in **(C-4)**) Then, **(C-9)** *GA subsumption* is applied, similar to $\{C\}$ *subsumption*. Lastly, **(C-10)** probabilistic *rule deletion* is used to limit the micro population size, `Rule_P`, in $\{P\}$ (see [29]). After this cycle completes, Phase I moves to the next instance, or returns to the first, if all training instances have been examined.

Phase I ends with **(D)** $\{P\}$ post-processing via QRF-based rule filtering [42] and subsumption-based compaction [30, 31] to yield a final $\{P\}$ aimed at presenting a diversity of high-quality candidate rules (see [29]). When including the newly proposed 'phase alternation' extension of HEROS, this filtering and compaction of $\{P\}$ occurs prior to every alternation back to Phase II.

2.1.3 Phase II - Model Optimization. Phase II combines candidate rules from $\{P\}$ (as building blocks) to assemble maximally accurate and compact rule-sets (i.e. models). Phase II disposes of solution 'fitness' in favor of NSGA-II-inspired [14] multi-objective fast non-dominated sorting, crowding distances, and front-rank/crowding-distance-based binary tournament parent selection [14, 29].

Models. An individual model includes a subset of unique rules from $\{P\}$ and a number of model parameters such as model *'balanced' accuracy*, (A_m), and *rule-set size* (S_m) (see [29]). During Phase II, new models have A_m evaluated after confirming that they do not already exist in the current *model population* $\{MP\}$ or the *offspring model population* $\{OMP\}$. When calculating A_m, rule-sets make a prediction on a training instance in a manner similar to a traditional LCS voting scheme (see [29]). Of note, HEROS rule-sets are unordered and do not use a *default rule* [2, 33].

Model Initialization. Phase II begins by **(E)** initializing a $\{MP\}$ of size (`Model_P`). Based on findings in [29], we adopt the (`model_pop_init`) strategy of *target accuracy initialization* which performed better than *random model initialization* (see [29]). Notably, when employing the newly introduced HEROS 'phase alternation' strategy, model initialization (like rule population initialization) occurs only once and does not repeat with alternations.

NSGA-II-Inspired EA. Following initialization, *NSGA-II-inspired optimization cycles* are applied for (*Model_I*) iterations. Each iteration, the cycle starts by applying **(F-1)** NSGA-II-style [14] fast non-dominated sorting of all models in $\{MP\}$ into ranked Pareto dominance fronts based on maximizing A_m and minimizing S_m. Then, crowding distances within each front are calculated and applied to encourage diversity during parent selection and deletion. Visuals can be found in Figure 1 and the appendix of [29].

Next, **(F-2)** the GA generates offspring models while the size of $\{OMP\}$ is less than (`Model_P` * `new_gen`). First, the GA applies NSGA-II-style binary tournament selection choosing two parent models from $\{MP\}$ [14]. Second, up to 3 offspring models are generated for each pair of parents via probabilistic application of the genetic operations (`p_merge`, `p_cross`, `p_mut`). Whenever a new offspring is a duplicate, the GA makes a finite number of attempts to initialize a novel random model (see [29]). Next, **(F-3)** $\{OMP\}$ is combined with $\{MP\}$ and another round of **(F-4)** non-dominated model sorting and crowding distance calculation is conducted. Then, **(F-5)** NSGA-II-style deletion [14] is applied to construct the next generation of $\{MP\}$ by selecting the best `Model_P` models according to ranked fronts. When a front is reached where all models on that front can't be added to $\{MP\}$, we add the models with the highest crowding distance until `Model_P` is reached.

Best Model Selection. Once $\{MP\}$ training has ended, HEROS yields a set of viable trained models on its non-dominated front. HEROS offers two strategies (**G-1** or **G-2**) for selecting a 'best' model: *model ranking* '**-D**' or *evaluate all models on front* '**-F**'. The former selects the single model with the largest training A_m (with larger coverage and smaller S_m as tie-breakers). This method ignores all other non-dominated models and is more susceptible to over-fitting. Method '**-F**' evaluates all non-dominated models using the *testing data* and reports the best model using the following strategy; starting with model with highest training accuracy, replace the current best model if the new one (1) has > balanced testing accuracy and ≥ testing coverage, or (2) has ≥ balanced testing accuracy, ≥ testing coverage, and < S_m. The **-F** strategy is 'advantaged' as it's akin to training multiple ML algorithms on the same training data and reporting the single algorithm's model that performed best on the testing data. While in this paper, the fairest comparisons to other algorithms use HEROS **-D**, HEROS **-F** is also used to illustrate how equally accurate, but more compact models can be found by considering all trained models on the non-dominated front.

Prediction Array. After training, HEROS makes class predictions on evaluation data using a prediction array which adopts a strategy similar to random forest [24] when multiple rules match a given instance (see [29]).

2.2 Phase Alternation

The first new HEROS extension is a simple 'phase alternation' strategy inspired by RBMLs that have differently focused on either fuzzy rules or regression tasks [23, 26, 35]. Original HEROS would run Phase I for `Rule_I` iterations and then subsequently run Phase II for `Model_I` iterations. The newly proposed HEROS alternation scheme simply rotates between Phase I and Phase II training iterations, with the number of alternations controlled by the hyperparameter '`alternate`' (a default of 5 used in this study). Allocated training iterations for both phases (i.e. `Rule_I` and `Model_I`) are divided evenly among alternations. Therefore, Phase I will run for $\frac{\texttt{Rule\_I}}{\texttt{alternate}}$ iterations before switching to Phase II, i.e., **(C/D)** → **(F)**. Of note, the first alternation only, Phase II will begin with model initialization, i.e., **(E/F)**. Then, after $\frac{\texttt{Model\_I}}{\texttt{alternate}}$ iterations, Phase II returns to Phase I, i.e., (**(F)** → **(C/D)**). This continues until the training iterations of both phases are exhausted. In the last alternation, any

indivisible resources are entirely used up by Phase I to ensure the algorithm ends with phase II learning iterations.

Notably, rules can be deleted within Phase I that had made it into models of Phase II in a prior alternation. Thus, all models save their own copy of respective rule objects so that they persist even if the original rule was deleted from $\{P\}$. This is done because immediate removal of a rule from a model may fundamentally change the model's performance and require re-evaluation, and such rules may still confer an advantage to a model even if deleted from $\{P\}$.

2.3 Tree-based Rule Initialization

The second new HEROS extension **(B)**, is a tree-based rule-population initialization strategy that rapidly trains a diversity of decision trees which are decomposed into IF:THEN rules to initialize the HEROS's $\{P\}$. This strategy is meant to boost Phase I rule discovery by (A) rapidly capturing initial associations in the dataset, and (B) providing greater rule diversity for the GA to evolve in **(C–7)**. It's also meant to counterbalance the drawbacks of phase alternation where early phase II iterations may only have access to an immature $\{P\}$ of rule 'building blocks', that may not have discovered any or all 'ideal' rules, or iterated through all training instances during **(C)**. This could 'waste' early Phase II training iterations that could be more effectively utilized with a more mature $\{P\}$. Thus, we expect tree initialization to benefit HEROS with or without alternation in datasets where decision trees can effectively capture some of the underlying associations.

Specifically, 8 random forest (RF) [7] models are trained (training data only) using scikit-learn's *RandomForestClassifier*, varying the *max_depth* hyperparameter from 1-7 and *None*, using 10 *estimators* (i.e. trees), and defaults for other hyperparameters (including *bootstrapping*) yielding 80 trees. All trees were decomposed to HEROS-style rules by traversing unique paths through a tree to leaf nodes. Since the RF implementation was only designed for quantitative features, we added temporary one-hot encoding for all categorical features prior to RF training. Since binary, one-hot encoded features don't have ranges, we use a default threshold value of 0.5 along with the direction of the split (either $\geq$ or $\leq$) to determine if the category was present. Then we mapped the one-hot encoded binary features back to their original categorical feature states within rules. Such features were only added to the specified features of a rule if the state was asserted in a tree's split (e.g. hair = black), otherwise that feature was left unspecified in the resulting rule (e.g. hair != black). For specified quantitative features, rules start with an infinite range [-inf, inf] which is tightened based on the condition of a given decision tree split e.g. [-inf, 23.5], for a tree branch specifying < 23.5. Next, all rules decomposed from trees were checked for redundancy based on rule conditions to form a unique set of rules. Lastly, before Phase I begins, all rules undergo batch evaluation, action assignment, checking for Pareto non-domination, and fitness update (like in **(C–4)**), before being added to $\{P\}$.

2.4 Algorithms for Comparison

This study compares HEROS algorithm configurations to two well-known RBML algorithms: RIPPER [11] and BioHEL [1]. These exemplify supervised classification RBMLs that focus on finding compact, interpretable models, providing excellent standards of comparison for both predictive performance and model compactness.

2.4.1 RIPPER. A widely used greedy rule-learning algorithm that employs a separate-and-conquer method to incrementally develop rule-sets [11]. First, RIPPER partitions the training data into a 'grow' vs. 'prune' set. Then it greedily generates a rule, covering positive class instances from the 'grow' set, that maximizes information gain until the rule is 100% accurate. Then this rule is pruned by stepping through all conditions and removing any that maintain or improve 'prune' set performance, adding it to the model. This process repeats until no positive examples are left or the rules become too inaccurate. Lastly, a global optimization evaluates and replaces redundant or inefficient rules within the entire rule-set.

RIPPER's focus on 'positive' instances means that it relies on a 'default' rule predicting a default class in the absence of other matching rules. As a result, predictions for default-class instances are not supported by explicit rule evidence, limiting interpretability. Further, RIPPER's ability to incorporate complex relationships is limited by its reliance on heuristic pruning and extensive local optimization. In domains that are exceedingly complex or heterogeneous, these constraints may lead to inadequate coverage of the input space and reduced predictive performance, similar to what might be expected for decision trees [18, 37].

2.4.2 BioHEL. An 'evolutionary' RBML with similarities to RIPPER, but employs different evaluation and processing heuristics for iterative rule discovery [1]. BioHEL repeatedly evolves single rules, using a GA, removes training instances covered by that rule, and then repeats the process for the remaining data. This separate-and-conquer strategy yields concise rule-sets and has been successfully applied to large-scale biomedical problems [1]. BioHEL's evolutionary search operates at the level of individual rules and is driven by a weighted multi-objective fitness function that balances rule accuracy and rule simplicity (i.e. minimizing the number of features specified in a rule). Of note, BioHEL uses an automatic weight adjustment heuristic proposed for its predecessor, GAssist [4]. The greedy removal of covered instances promotes rapid convergence and compact models, but also introduces an implicit assumption that previously discovered rules are globally optimal. In noisy, chaotic, or heterogeneous environments, its possible that initial poor rules may skew subsequent learning and constrain robustness.

BioHEL also offers a Rule Post-processing Engine (RPE), which adds further refinement operations subsequent to rule induction [17]. The RPE assesses the whole rule-set and uses techniques such as rule pruning, reordering, and simplification to diminish redundancy and enhance generality. While this post-processing step can yield more compact and coherent rule-sets, it does not alter BioHEL's reliance on a default-class (like RIPPER), also limiting interpretability in those cases. In this study, we evaluate BioHEL both with and without RPE.

2.5 Experimental Evaluation

This section describes the experimental design used to evaluate and compare the proposed HEROS extensions, including algorithm configurations, datasets, and evaluation protocol.

Table 2: Simulated GAMETES (A-F) and MUX (6-70) datasets.

Data Name	Tot. Feat.	Pred. Feat.	Inst.	Max. Acc.	Nature of Underlying Association	Optimal Rule Count
A	100	4	1600	≈ 0.94	additive univariate	Unknown
B	100	1	1600	≈ 0.84	univariate	≳ 3
C	100	2	1600	≈ 0.82	2-way epistasis	≳ 9
D	100	4	1600	≈ 0.70	2-het. x 2-way epistasis	≳ 18
E	100	4	1600	≈ 0.65	4-het. x univariate	≳ 12
F	100	3	1600	≈ 0.65	3-way epistasis	≳ 27
MUX6	6	6	500	1	4-het. x 3-way epistasis	8
MUX11	11	11	5000	1	8-het. x 4-way epistasis	16
MUX20	20	20	10000	1	16-het. x 5-way epistasis	32
MUX37	37	37	10000	1	32-het. x 6-way epistasis	64
MUX70	70	70	20000	1	64-het. x 7-way epistasis	128

2.5.1 Simulated Benchmark Datasets. This study benchmarks all algorithms using a diverse and challenging set of datasets used to demonstrate that HEROS outperformed ExSTraCS [29]. Table 2 provides a summary of the characteristics these 11 benchmarks which are further discussed in [29]. In summary, these include 6 genomic datasets (A-F) simulated with GAMETES [40, 41] as noisy datasets with different underlying association types and noise levels. Also, 5 increasingly complex MUX datasets with completely clean signals are included (MUX -6, 11, 20, 37,and 70). In all datasets the ground-truth predictive features are known, and in the MUX datasets the 'ideally' compact rule sets are known. The optimal rule counts given assume a rule-set that does not use default rules. Later, when evaluating RIPPER and BioHEL, which rely on default rules) we take their smaller ideal rule count into account.

2.5.2 Algorithm Configurations. In total, this study examined 9 algorithm configurations; 6 for HEROS, 2 for BioHEL, and 1 for RIPPER. Three implementations of HEROS were run including two that incorporate our proposed expansions: (1) original HEROS (i.e. '**HEROS**'), (2) HEROS with the addition of tree-based rule initialization only (i.e. '**HEROS-Tree**'), and (3) HEROS with both tree initialization and phase alternation (i.e. '**HEROS-Tree-Alt.**'). Additionally, for each implementation, HEROS was run using both available strategies to select a 'best model' from its final non-dominated model front; '**-D**', and '**-F**', e.g. **HEROS-F**. Original HEROS previously demonstrated that better performing models were generally found with '**-F**', however this affords the algorithm the added advantage of reporting the model with the best testing performance across the restricted set of non-dominated models identified at the end of training [29]. We also ran RIPPER as a greedy baseline approach. It offers a computationally efficient reference point for evaluating interpretability-oriented rule learning, despite the absence of explicit global model refinement or evolutionary optimization. And lastly we ran BioHEL as a hybrid evolutionary RBML that is also capable of generating compact interpretable rules sets and was designed for real world biomedical problems. BioHEL was run alone or in combination with its rule post-processing engine (RPE), referenced in the results as either '**BioHEL**' or '**BioHEL-RPE**'.

2.5.3 Evaluation Protocol. The original HEROS algorithm was developed to be 'noise agnostic' and have the capability to tackle different problem types without relying on dataset-specific hyperparameter tuning. Thus in the present study for each of the above algorithm configurations, all algorithms were evaluated using fixed hyperparameter configurations across all datasets. For RIPPER and BioHEL, we initially ran both algorithms using their default hyperparameters (results not shown). While this study applies default HEROS hyperparameters [29] (see Table 1) this study has sought to select more computationally expensive and more HEROS-comparable hyperparameter settings for RIPPER and BioHEL, although there is not clear way to ensure this is the case given fundamental algorithm differences. Tables giving selected RIPPER and BioHEL hyperparameters are given in supplementary materials. RIPPER and BioHEL results with default hyperparameters (not shown) were largely comparable to those presented below.

All trials were conducted across the GAMETES and MUX benchmarks in Table 2. Each dataset was pre-divided into 10 stratified cross validation (CV) folds to ensure all algorithm trainings and evaluations were conducted on the same partitions. Further, all algorithm trainings were repeated with 20 random seed runs. Thus with 11 benchmark datasets, 20 random seeds, 10-fold CV, and 9 algorithm configurations, a total of 19800 algorithm training and testing evaluations were conducted.

Performance is evaluated using the following metrics: 'balanced' testing accuracy, testing coverage, rule count, and run time (in minutes). Balanced accuracy is the average of sensitivity and specificity, able to take class imbalance into consideration. Testing coverage is the proportion of testing instances that matched a rule in the model's rule-set. For RIPPER and BioHEL we give the proportion of testing instances that matched a non-default rule, since the default rule makes it so that all instances are naturally covered (i.e. coverage = 1.0). Using this alternative definition allows us to inspect the degree which these algorithms rely on their default rules for predictions. Rule count is simply the size of the rule-set. Of note, while individual RBML rules of all algorithms examined are inherently interpretable as previously demonstrated [1, 11, 29, 42], the size of the rule-set serves an indicator of model interpretability (where more compact rule sets are more desirable), with the caveat that HEROS is expected to require at least 2 times the number of rules to ideally capture associations than RIPPER and BioHEL, which each rely on a default rule to cover one of the two classes in all experiments. While this simplifies rule sets, it leaves instances predicted by the default class without interpretable justification beyond having not been matched by other rules in the rule-set. For the MUX datasets, we also examine 'ideal count' which gives the proportion of the 200 (i.e. 10-fold CV * 20 seeds) runs for that algorithm and MUX dataset, where the known ideally compact set of rules was identified by the algorithm. For RIPPER and BioHEL, we assume a default rule in the evaluation of what the ideally compact set size should be.

Statistical significance was assessed with the Wilcoxon rank-sum test comparing individual algorithm configurations to **HEROS-D** as our statistical baseline with and without Benjamini Hochberg FDR multiple testing corrections ($p < 0.05$). All experiments were conducted on a Linux-based high-performance computer cluster. This study used HEROS v0.2.5 as 'original' HEROS to fix identified rule discovery inefficiencies in v0.2 used in [29]. This accounts for higher HEROS rule counts after the allotted Phase II iterations observed in this study vs. [29] (analysis not shown). All HEROS versions and analysis scripts are available at https://github.com/UrbsLab/heros. All simulated datasets are available upon request.

Table 3: Algorithm Comparison on GAMETES Datasets

Data	Algorithm Configuration	Testing Accuracy	Testing Coverage	Rule Count	Run Time (Min.)
A	HEROS-**D**	0.908(0.012)	1.000(0.000)	172.0(27.9)	35.4(1.3)
	HEROS-Tree **-D**	0.911(0.011)	1.000(0.000)	165.8(16.5)	37.3(1.4)↑*
	HEROS-Tree-Alt.-**D**	0.924(0.003)↑*	1.000(0.000)	170.7(24.5)	39.5(0.9)↑*
	HEROS-**F**	0.929(0.006)↑*	1.000(0.000)	86.7(23.1)↓*	35.4(1.3)
	HEROS-Tree-**F**	0.931(0.005)↑*	1.000(0.000)	85.3(20.7)↓*	37.3(1.4)↑*
	HEROS-Tree-Alt.-**F**	0.942(0.003)↑*	1.000(0.000)	57.5(13.7)↓*	39.5(0.9)↑*
	RIPPER	0.906(0.021)↓*	0.449(0.005)↓*	14.0(2.4)↓*	<0.1(0.0)↓*
	BioHEL	0.857(0.026)↓*	0.457(0.028)↓*	55.7(2.0)↓*	29.1(5.7)↓*
	BioHEL-RPE	0.857(0.026)↓*	0.456(0.028)↓*	54.8(2.2)↓*	29.2(5.7)↓*
B	HEROS-**D**	0.826(0.002)	0.998(0.001)	70.2(14.2)	28.3(1.4)
	HEROS-Tree-**D**	0.827(0.002)	0.998(0.001)	70.5(21.3)	31.1(1.5)↑*
	HEROS-Tree-Alt.-**D**	0.827(0.002)	0.999(0.001)	68.2(13.5)	31.0(1.1)↑*
	HEROS-**F**	0.829(0.002)↑*	0.999(0.001)↑*	33.7(10.9)↓*	28.3(1.4)
	HEROS-Tree-**F**	0.829(0.002)↑*	0.998(0.001)	34.8(8.4)↓*	31.5(1.5)↑*
	HEROS-Tree-Alt.-**F**	0.831(0.001)↑*	0.999(0.000)↑*	23.8(7.9)↓*	31.0(1.1)↑*
	RIPPER	0.818(0.031)↓*	0.457(0.005)↓*	5.7(2.5)↓*	<0.1(0.0)↓*
	BioHEL	0.635(0.035)↓*	0.327(0.042)↓*	75.6(1.6)↑*	45.1(7.4)↑*
	BioHEL-RPE	0.635(0.035)↓*	0.329(0.042)↓*	75.4(1.7)↑*	45.1(7.4)↑*
C	HEROS-**D**	0.798(0.004)	0.993(0.003)	140.9(35.4)	39.1(2.1)
	HEROS-Tree-**D**	0.799(0.004)	0.993(0.003)	132.9(45.3)	39.8(3.0)
	HEROS-Tree-Alt.-**D**	0.795(0.004)↓	0.995(0.002)↑	219.1(61.3)↑*	47.7(3.9)↑*
	HEROS-**F**	0.803(0.004)↑*	0.994(0.002)↑*	81.3(34.1)↓*	39.1(2.1)
	HEROS-Tree-**F**	0.805(0.004)↑*	0.994(0.003)	79.9(35.1)↓*	39.8(3.0)
	HEROS-Tree-Alt.-**F**	0.806(0.004)↑*	0.995(0.002)↑*	112.1(36.4)↓*	47.7(3.9)↑*
	RIPPER	0.541(0.100)↓*	0.419(0.025)↓*	2.4(3.0)↓*	<0.1(0.0)↓*
	BioHEL	0.596(0.035)↓*	0.318(0.036)↓*	81.1(2.3)↓*	81.1(2.3)↑*
	BioHEL-RPE	0.596(0.036)↓*	0.32(0.036)↓*	80.9(2.3)↓*	51.1(6.3)↑*
D	HEROS-**D**	0.666(0.007)	1.000(0.000)	310.2(28.7)	40.9(3.3)
	HEROS-Tree-**D**	0.665(0.006)	1.000(0.001)	313.4(35.5)	42.8(2.5)↑*
	HEROS-Tree-Alt.-**D**	0.668(0.007)	1.000(0.000)	280.4(40.6)↓*	44.7(4.4)↑*
	HEROS-**F**	0.682(0.004)↑*	1.000(0.000)	153.2(23.3)↓*	40.9(3.3)
	HEROS-Tree-**F**	0.680(0.005)↑*	1.000(0.000)	166.4(26.9)↓*	42.8(2.5)↑*
	HEROS-Tree-Alt.-**F**	0.687(0.007)↑*	1.000(0.000)	150.4(33.0)↓*	44.7(4.4)↑*
	RIPPER	0.511(0.032)↓*	0.417(0.022)↓*	1.8(1.1)↓*	<0.1(0.0)↓*
	BioHEL	0.525(0.039)↓*	0.277(0.038)↓*	89.6(2.2)↓*	60.5(7.7)↑*
	BioHEL-RPE	0.526(0.039)↓*	0.28(0.038)↓*	89.5(2.1)↓*	60.5(7.7)↑*
E	HEROS-**D**	0.608(0.007)	0.999(0.001)	130.5(22.7)	39.4(1.5)
	HEROS-Tree-**D**	0.607(0.008)	1.000(0.000)	138.6(22.3)	43.5(2.1)↑*
	HEROS-Tree-Alt.-**D**	0.606(0.005)	1.000(0.000)↑*	113.0(12.8)↓*	42.0(1.9)↑*
	HEROS-**F**	0.634(0.005)↑*	1.000(0.001)↑	63.0(14.9)↓*	39.4(1.5)
	HEROS-Tree-**F**	0.631(0.004)↑*	1.000(0.000)↑*	69.5(15.8)↓*	43.5(2.1)↑*
	HEROS-Tree-Alt.-**F**	0.632(0.004)↑*	1.000(0.000)↑*	51.3(11.0)↓	42.0(1.9)↑*
	RIPPER	0.563(0.036)↓*	0.44(0.009)↓*	3.7(1.4)↓	<0.1(0.0)↓
	BioHEL	0.523(0.034)↓*	0.258(0.040)↓*	90.8(1.8)↓*	61.7(7.6)↑*
	BioHEL-RPE	0.523(0.034)↓*	0.261(0.040)↓*	90.6(1.8)↓*	61.7(7.6)↑*
F	HEROS **-D**	0.502(0.010)	0.996(0.002)	263.3(40.0)	56.5(2.3)
	HEROS-Tree-**D**	0.503(0.008)	0.996(0.001)	295.0(54.2)	56.6(4.2)
	HEROS-Tree-Alt.-**D**	0.498(0.007)	0.998(0.001)↑*	209.6(39.2)↓*	57.5(1.4)↑*
	HEROS-**F**	0.522(0.011)↑*	0.997(0.002)↑*	170.3(37.3)↓*	56.5(2.3)
	HEROS-Tree-**F**	0.525(0.007)↑*	0.998(0.001)↑*	184.2(48.6)↓*	56.6(4.2)
	HEROS-Tree-Alt.-**F**	0.523(0.009)↑*	0.999(0.001)↑*	108.0(19.6)↓*	57.5(1.4)↑*
	RIPPER	0.498(0.021)	0.432(0.024)↓*	1.3(0.5)↓*	<0.1(0.0)↓*
	BioHEL	0.499(0.034)	0.253(0.039)↓*	92.4(2.1)↓*	61.0(7.9)↑*
	BioHEL-RPE	0.499(0.034)	0.256(0.039)↓*	92.3(2.1)↓*	61.0(7.8)↑*

Table 4: Algorithm Comparison on MUX Datasets

Data	Algorithm Configuration	Testing Accuracy	Testing Coverage	Rule Count	Run Time (Min.)	Ideal Count
6	HEROS-**D**	1.000(0.000)	1.000(0.000)	8.0(0.0)	2.0(0.0)	200/200
	HEROS-Tree **-D**	1.000(0.000)	1.000(0.000)	8.0(0.0)	2.0(0.1)	200/200
	HEROS-Tree-Alt. **-D**	1.000(0.000)	1.000(0.000)	8.0(0.0)	2.2(0.0)↑*	200/200
	HEROS-**F**	1.000(0.000)	1.000(0.000)	8.0(0.0)	2.0(0.0)	200/200
	HEROS-Tree-**F**	1.000(0.000)	1.000(0.000)	8.0(0.0)	2.0(0.1)	200/200
	HEROS-Tree-Alt.-**F**	1.000(0.000)	1.000(0.000)	8.0(0.0)	2.2(0.0)↑*	200/200
	RIPPER	1.000(0.000)	0.586(0.032)↓*	5.4(0.9)↓*	<0.1(0.0)↓*	27/200 ↓
	BioHEL	1.000(0.000)	0.500(0.000)↓*	6.2(1.1)↓*	1.3(0.3)↓*	48/200↓
	BioHEL-RPE	1.000(0.000)	0.500(0.000)↓*	4.8(0.9)↓*	1.3(0.3)↓*	68/200↓
11	HEROS-**D**	0.992(0.006)	0.998(0.002)	19.4(1.5)	59.7(1.5)	89/200
	HEROS-Tree-**D**	0.990(0.005)	0.996(0.003)↓*	19.6(2.0)	62.2(6.7)↑*	83/200 ↓
	HEROS-Tree-Alt.-**D**	0.995(0.004)↑	0.997(0.002)	19.9(2.0)	68.1(1.6)↑*	77/200 ↓
	HEROS-**F**	0.994(0.005)↑*	0.998(0.002)	18.3(1.0)↓*	59.7(1.5)	97/200 ↑
	HEROS-Tree-**F**	0.991(0.004)	0.996(0.002)↓*	18.2(1.2)↓*	62.2(6.7)↑*	88/200 ↓
	HEROS-Tree-Alt.-**F**	0.996(0.003)↑*	0.997(0.001)	18.4(1.3)↓*	68.1(1.6)↑*	93/200 ↑
	RIPPER	1.000(0.000)↑*	0.563(0.013)↓*	11.3(1.2)↓*	<0.1(0.0)↓*	0/200 ↓
	BioHEL	1.000(0.000)↑*	0.500(0.000)↓*	13.2(1.9)↓*	33.5(6.1)↓*	3/200 ↓
	BioHEL-RPE	1.000(0.000)↑*	0.500(0.000)↓*	9.3(1.2)↓*	33.5(6.2)↓*	54/200 ↓
20	HEROS-**D**	0.888(0.014)	0.991(0.004)	115.4(16.3)	151.9(5.7)	0/200
	HEROS-Tree-**D**	0.891(0.014)	0.990(0.005)	112.5(14.8)	174.5(6.1)↑*	0/200
	HEROS-Tree-Alt.-**D**	0.940(0.009)↑*	0.992(0.003)	119.7(15.4)	171.4(7.2)↑*	0/200
	HEROS-**F**	0.894(0.013)↑*	0.992(0.004)↑*	104.2(14.4)↓*	151.9(5.7)	0/200
	HEROS-Tree-**F**	0.897(0.011)↑*	0.990(0.005)	101.0(11.0)↓*	174.5(6.1)↑*	0/200
	HEROS-Tree-Alt.-**F**	0.943(0.007)↑*	0.992(0.003)	108.7(12.1)	171.4(7.2)↑*	0/200
	RIPPER	0.996(0.032)↑*	0.533(0.007)↓*	25.12(3.1)↓*	0.1(0.0)↓*	0/200
	BioHEL	1.000(0.000)↑*	0.500(0.000)↓*	31.4(3.3)↓*	172.5(14.5)↑*	0/200
	BioHEL-RPE	1.000(0.000)↑*	0.500(0.000)↓*	16.9(1.1)↓*	172.5(14.5)↑*	55/200 ↑
37	HEROS-**D**	0.562(0.004)	1.000(0.000)	211.4(14.1)	184.7(5.4)	0/200
	HEROS-Tree-**D**	0.562(0.005)	1.000(0.000)	215.1(9.2)	187.7(10.1)↑*	0/200
	HEROS-Tree-Alt.-**D**	0.568(0.005)↑*	1.000(0.000)	238.7(18.5)↑*	205.2(7.8)↑*	0/200
	HEROS-**F**	0.583(0.004)↑*	1.000(0.000)	103.8(16.3)↓*	184.7(5.4)	0/200
	HEROS-Tree-**F**	0.583(0.004)↑*	1.000(0.000)	104.7(14.9)↓*	187.7(10.1)↑*	0/200
	HEROS-Tree-Alt.-**F**	0.589(0.004)↑*	1.000(0.000)	108.4(13.7)↓*	205.2(7.8)↑*	0/200
	RIPPER	0.799(0.184)↓*	0.527(0.011)↓*	43.1(29.0)↓*	0.2(0.2)↓*	0/200
	BioHEL	0.995(0.005)↑*	0.504(0.004)↓*	61.3(6.0)↓*	314.9(55.4)↑*	0/200
	BioHEL-RPE	0.997(0.004)↑*	0.502(0.003)↓*	39.1(6.2)↓*	315.0(55.4)↑*	19/200 ↑
70	HEROS-**D**	0.523(0.003)	1.000(0.000)	329.3(14.9)	423.2(61.8)	0/200
	HEROS-Tree-**D**	0.524(0.003)	1.000(0.000)	323.4(20.5)	465.3(15.6)↑*	0/200
	HEROS-Tree-Alt.-**D**	0.525(0.002)↑*	1.000(0.000)	363.2(24.3)↑*	447.8(64.0)↑*	0/200
	HEROS-**F**	0.536(0.002)↑*	1.000(0.000)	145.0(28.8)↓*	423.2(61.8)	0/200
	HEROS-Tree-**F**	0.537(0.002)↑*	1.000(0.000)	146.2(21.6)↓*	465.3(15.6)↑*	0/200
	HEROS-Tree-Alt.-**F**	0.539(0.002)↑*	1.000(0.000)	128.3(23.6)↓*	447.8(64.0)↑*	0/200
	RIPPER	0.508(0.016)↓*	0.528(0.017)↓*	3.9(4.2)↓*	0.1(0.1)↓*	0/200
	BioHEL	0.973(0.007)↑*	0.512(0.004)↓*	133.0(8.6)↓*	1519.5(108.3)↑*	0/200
	BioHEL-RPE	0.94(0.008)↑*	0.512(0.004)↓*	118.0(11.0)↓*	1520.2(108.4)↑*	0/200

3 Results and Discussion

Tables 3 and 4 compare the mean (standard deviation) results of all algorithm configurations for the GAMETES and MUX datasets, respectively. In both, original HEROS using '-D' model selection (i.e. **HEROS-D**) is shaded as the baseline of statistical comparison for all other configurations. The best performing algorithm (based on testing accuracy) is highlighted in yellow (with 'ideal count' as a tie breaker for MUX datasets). Significant performance differences are indicated by ↑ or ↓, respectively and '*' indicates significance after FDR multiple testing corrections (p-vals. in supp. mats.). Table 4, 'ideal count' is given as a proportion of successes out of 200, and arrows indicate any differences from **HEROS-D** baseline.

Focusing first on RBML results in noisy GAMETES datasets (Table 3), we summarize key observations: (1) in datasets A-E, all HEROS algorithm configurations achieved higher testing accuracies than RIPPER or BioHEL (2) HEROS with '**-F**' yielded similar or smaller rule-sets than BioHEL in datasets A, B, C, and E despite BioHEL having the advantage of a default rule (3) no algorithms performed particularly well on the high noise, 3-way interaction in dataset F, however, of these, '**HEROS-Tree-F**' performed best, (4) as expected, HEROS configurations using '**-F**' generally yielded better accuracy and rule-set size than '**-D**', (5) the addition of tree initialization alone to HEROS had minimal impact on the end performance of the algorithm, (6) further addition of phase alternation to HEROS (i.e.'**HEROS-Tree-Alt.**') yielded similar, and sometimes better performance than '**HEROS**' or '**HEROS-Tree**' for both '**-D**' and '**-F**', (7) '**HEROS-Tree-Alt.-F**' yielded the best performance of all RBML configurations in datasets A-D, and only slightly less well than '**HEROS-F**' in dataset E or '**HEROS-Tree-F**' in dataset F, (8) HEROS configurations had consistently high testing coverage (at or near 1.0), while RIPPER and BioHEL are consistently below their expected ideal coverage of ≈ 0.5 coverage (given their default rule) on these balanced class datasets, (9) HEROS configurations yielded consistently lower SD values for testing accuracy suggesting better convergence, (10) the addition of RPE to BioHEL had minimal impact, (11) RIPPER ran, by far, fastest, and BioHEL and HEROS had similar run times in most analyses, and (12) RIPPER produced very small rule-sets when performing poorly.

Shifting to RBML results in clean MUX datasets (Table 4), we again summarize key observations: (1) overall, BioHEL and RIPPER yielded higher testing accuracies than HEROS configurations in MUX11-37 datasets, (2) BioHEL achieved dramatically higher testing accuracies than any other algorithms in the MUX37 and MUX70 datasets, (3) however, while all algorithms achieved 100% testing accuracy on MUX6, only the HEROS configurations consistently identified the ideally compact rule-set in all 200 runs, (4) additionally, on MUX11, while averaged performance of HEROS configurations were below RIPPER and BIOHEL, all HEROS configurations achieved a higher success rate in identifying ideally compact rule-sets, (5) again, RPE in BioHEL had minimal impact

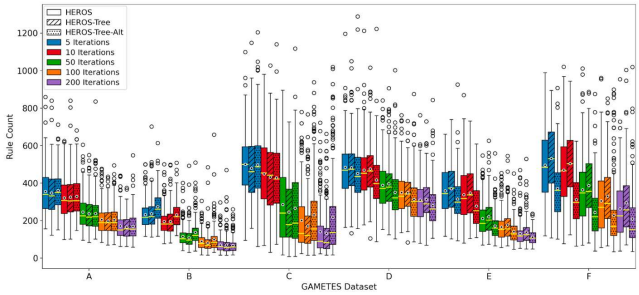


Figure 2: Rule counts of 'HEROS-D' algorithm configurations on GAMETES datasets across Phase II learning iteration checkpoints. Textures indicate algorithm configurations, and colors indicate learning checkpoints. Each box-plot includes 200 (i.e. 10-fold CV * 20 seeds) algorithm runs.

on performance, (6) both BioHEL and RIPPER yielded testing coverages much closer to their expected ideals of ≈ 0.5, (7) our experimental configurations of '**HEROS-Tree**' and '**HEROS-Tree-Alt.**' yielded similar or better performance than original HEROS for both '**-D**' and '**-F**', (8) again, HEROS configurations using '**-F**' generally yielded better performance than '**-D**', (9) again RIPPER ran, by far the fastest. Of note, in previous work on these same benchmarks, the MIC-style RBML, ExSTraCS (with $\nu = 1$) achieved similar testing accuracies to BioHEL on the MUX6-20, with marginally lower accuracy in MUX37, and a dramatically lower accuracy in MUX70, albeit with much larger rule-sets than BioHEL [29]. Further, both original HEROS and ExSTraCS (with $\nu = 10$) achieved almost identical testing accuracies to BioHEL in MUX6-37, but both dropped off in performance for MUX70.

We also compared HEROS configurations across Phase II checkpoints (i.e., after 5, 10, 50, 100, and 200 iterations). Figure 2 examines rule counts of different '**HEROS-D**' algorithm configurations on GAMETES datasets across checkpoints. Here we can observe the progressive reduction in rule-set size as all HEROS configurations converge on solutions. We observe similar performance convergence for both original '**HEROS**' and '**HEROS-Tree-Alt.**'. Complementary box plots for rule count and testing accuracy across MUX and GAMETES datasets using '**-D**' and '**-F**' are given in sup. mats. While not explicitly investigated here, preliminary testing suggested that adding *phase alternation* alone to HEROS led to worse performance at checkpoints (as would be expected).

While *tree initialization* alone had minimal impact on benchmark performance, the combination of tree initialization with *phase alternation* preserved or improved performance in contrast with original HEROS while also achieving the goal of adapting HEROS as an 'anytime algorithm'. Further, on noisy GAMETES datasets, HEROS algorithm configurations, especially '**HEROS-Tree-Alt.**' (**-D** and **-F**) consistently outperformed both RIPPER and BioHel. While HEROS algorithm configurations performed well for 6MUX and 11MUX, they were outperformed by RIPPER and (in particular) BioHEL in higher complexity MUX benchmarks.

Ultimately, the noise-agnostic flexibility of HEROS (with limited overfitting) seems to be earned at the expense of less efficiently solving clean problems of very high complexity. This is similar to previous observations comparing HEROS to accuracy-driven ExSTraCS [29], which also performed well on these MUX20 and MUX37 benchmarks. That study also found that HEROS performance on MUX20 and MUX37 could also be recovered by setting setting $\nu = 10$; forcing HEROS to perform as an accuracy-driven, rather than agnostic, approach [29]. Beyond their a larger focus on accuracy, both RIPPER and BioHEL also have the advantage of a default rule, which greatly simplifies rule search (particularly in binary class problems) given that roughly half as many rules need to be discovered and incorporated into the rule-set. However, while a default rule makes the rule-set more compact, it also adds ambiguity as to 'why' the default class was predicted on future instances. Overall, the results presented in this paper suggest that HEROS is very effective in both 'noisy', and moderately-complex 'clean', simulation problems. While further validation and benchmarking across a variety of real-world datasets is still need, we expect HEROS to offer effective classification modeling in diverse real-world problems that demand model transparency and interpetability.

4 Conclusions and Future Work

This study introduced *tree-based rule initialization* and *phase alternation* to HEROS; a promising, 'staged', two-phase evolutionary RMBL algorithm that separates rule from rule-set optimization using distinct non-weighted multi-objective optimizations. Our primary goal was to adapt the interpretable and noise-agnostic HEROS algorithm as an 'anytime algorithm' capable of broader algorithmic expansion and significantly increased practicality for real-world applications. Additionally, for the first time, this study compares HEROS and, indirectly, the ExSTraCS algorithm, to other well known RBMLs capable of yielding highly interpretable and compact rule-sets including RIPPER, and BioHEL. Results demonstrate that our proposed expansions successfully extend HEROS as an 'anytime algorithm' while improving or maintaining the empirical performance of the original HEROS algorithm. Further, our expanded HEROS implementation consistently outperformed both RIPPER and BioHEL on a diversity of more realistic 'noisy' simulated benchmark datasets as well as showed competitive performance on the clean MUX6 and MUX11 benchmarks. However, BioHEL, and to certain extent, RIPPER demonstrated superior performance on higher-dimensional, clean, MUX problems, likely due to their greater emphasis on accuracy and their use of a default rule to simplify search space convergence.

This work offers many future opportunities including; (1) strategies that provide feedback between HEROS phases, e.g., mutualistic co-evolution, (2) incorporation of BioHEL as a HEROS rule-set discovery operator to accelerate performance in clean but complex domains, (3) examining efficient optimization of tree-based rule initialization, (4) exploring optimization of the phase alternation hyperparameter, (5) HEROS extension to quantitative and survival outcomes (e.g.[48]), and (6) evaluating and improving HEROS scalability on a broader diversity of simulated and real-world datasets.

Acknowledgments

The study was supported by Cedars Sinai Health Sciences University and NIH grants R01 AI173095, U01 AG066833, P01 HL160471 and P30 AG073105. Special thanks to Felix Heptner, Lars Britz, Kia Kazemi-Nia, and reviewers for their manuscript feedback.

References

[1] Jaume Bacardit, Edmund K. Burke, and Natalio Krasnogor. 2009. Improving the scalability of rule-based evolutionary learning. *Memetic Computing* 1, 1 (March 2009), 55–67. doi:10.1007/s12293-008-0005-4

[2] Jaume Bacardit, Edmund K Burke, and Natalio Krasnogor. 2009. Improving the scalability of rule-based evolutionary learning. *Memetic Computing* 1, 1 (2009), 55–67.

[3] Jaume Bacardit and Natalio Krasnogor. 2009. A mixed discrete-continuous attribute list representation for large scale classification domains. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. 1155–1162.

[4] Jaume Bacardit Peñarroya. 2004. *Pittsburgh genetic-based machine learning in the data mining era: representations, generalization, and run-time.* Ph. D. Dissertation. Universitat Ramon Llull.

[5] Ester Bernadó-Mansilla and Josep M Garrell-Guiu. 2003. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary computation* 11, 3 (2003), 209–238.

[6] Lashon B. Booker, David E. Goldberg, and John H. Holland. 1989. Classifier systems and genetic algorithms. *Artificial intelligence* 40, 1-3 (1989), 235–282.

[7] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[8] Larry Bull. 2005. Foundations of learning classifier systems. In *Foundations of Learning Classifier Systems*, Larry Bull (Ed.). Springer, 1–25.

[9] Clément Bénard, Gérard Biau, Sébastien Da Veiga, and Erwan Scornet. 2021. SIRUS: Stable and Interpretable RUle Set for classification. *Electronic Journal of Statistics* 15 (01 2021), 427–505. doi:10.1214/20-EJS1792

[10] D. R. Carvalho and Alex A. Freitas. 2004. A hybrid decision tree/genetic algorithm method for data mining. *Information Sciences* 163, 1-3 (2004), 13–35.

[11] William W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*. William W Cohen, 115–123.

[12] Kalyanmoy Deb. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons.

[13] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[14] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[15] P-S Deng and EG Tsacle. 2000. Coupling genetic algorithms and rule-based systems for complex decisions. *Expert Systems with Applications* 19, 3 (2000), 209–218.

[16] María A Franco, Natalio Krasnogor, and Jaume Bacardit. 2010. Speeding up the evaluation of evolutionary learning systems using GPGPUs. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. 1039–1046.

[17] Maria A. Franco, Natalio Krasnogor, and Jaume Bacardit. 2012. Post-processing operators for decision lists. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12*. Philadelphia, Pennsylvania, USA, 847. doi:10.1145/2330163.2330281

[18] Bishwamittra Ghosh, Dmitry Malioutov, and Kuldeep S. Meel. 2022. Efficient learning of interpretable classification rules. *Journal of Artificial Intelligence Research* 74 (2022), 1249–1293.

[19] Michael Heider. 2025. *Disentangling the model selection tasks for improved explainability in a rule-based machine learning system.* Doktorarbeit. Universität Augsburg.

[20] Michael Heider, Maximilian Krischan, Roman Sraj, and Jörg Hähner. 2024. Exploring Self-Adaptive Genetic Algorithms to Combine Compact Sets of Rules. In *2024 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[21] Michael Heider, Helena Stegherr, David Pätzel, Roman Sraj, Jonathan Wurth, Benedikt Volger, and Jörg Hähner. 2022. Approaches for Rule Discovery in a Learning Classifier System. In *Proceedings of the 14th International Joint Conference on Computational Intelligence - ECTA*. INSTICC, SciTePress, Setúbal, Portugal, 39–49. doi:10.5220/0011542000003332

[22] Michael Heider, Helena Stegherr, David Pätzel, Roman Sraj, Jonathan Wurth, Benedikt Volger, and Jörg Hähner. 2023. Discovering Rules for Rule-Based Machine Learning with the Help of Novelty Search. *SN Computer Science* 4, 6 (2023), 778. doi:10.1007/s42979-023-02198-x

[23] Michael Heider, Helena Stegherr, Roman Sraj, David Pätzel, Jonathan Wurth, and Jörg Hähner. 2023. SupRB in the context of rule-based machine learning methods: A comparative study. *Applied Soft Computing* 147 (2023), 110706. doi:10.1016/j.asoc.2023.110706

[24] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1. IEEE, 278–282.

[25] Giles Hooker, Lucas Mentch, and Siyu Zhou. 2021. Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance. *Statistics and Computing* 31, 6 (2021), 82.

[26] Hisao Ishibuchi, Tomoharu Nakashima, and Tetsuya Kuroda. 2000. A hybrid fuzzy GBML algorithm for designing compact fuzzy rule-based classification systems. In *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063)*, Vol. 2. IEEE, 706–711.

[27] Patrick Knab, Sascha Marton, Udo Schlegel, and Christian Bartelt. 2025. Which lime should i trust? concepts, challenges, and solutions. In *World Conference on Explainable Artificial Intelligence*. Springer, 28–52.

[28] Yan Li, Shuzhi Ge, et al. 2014. A modified decision tree algorithm based on genetic algorithm. *The Scientific World Journal* 2014 (2014), 1–9.

[29] Gabriel Lipschutz-Villa, Harsh Bandhey, Ruonan Yin, Malek Kamoun, and Ryan J. Urbanowicz. 2025. Rule-based Machine Learning: Separating Rule and Rule-Set Pareto-Optimization for Interpretable Noise-Agnostic Modeling. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '25)*. ACM.

[30] Yi Liu, Will N Browne, and Bing Xue. 2019. Absumption to complement subsumption in learning classifier systems. In *Proceedings of the genetic and evolutionary computation conference*. 410–418.

[31] Yi Liu, Will N Browne, and Bing Xue. 2021. A comparison of learning classifier systems' rule compaction algorithms for knowledge visualization. *ACM Transactions on Evolutionary Learning and Optimization* 1, 3 (2021), 1–38.

[32] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. 2019. Interpretable machine learning: Definitions, methods, and applications. *Proceedings of the National Academy of Sciences* 116, 44 (2019), 22071–22080.

[33] Jaume Bacardit Peñarroya. 2004. *Pittsburgh genetic-based machine learning in the data mining era: representations, generalization, and run-time.* Ph. D. Dissertation. Universitat Ramon Llull.

[34] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215.

[35] Roman Sergienko and Eugene Semenkin. 2013. Michigan and Pittsburgh methods combination for fuzzy classifier design with coevolutionary algorithm. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, 3252–3259.

[36] Ivor W. Tsang et al. 2014. Evolving decision trees with beam search-based initialization and evolutionary operators. *Information Sciences* 258 (2014), 340–356.

[37] Ryan J. Urbanowicz. [n. d.]. Learning Classifier Systems: An Introduction and Roadmap. https://sites.google.com/site/ryanurbanowicz/learning-classifier-systems Accessed 2026-01-05.

[38] Ryan J Urbanowicz, Gediminas Bertasius, and Jason H Moore. 2014. An extended michigan-style learning classifier system for flexible supervised learning, classification, and data mining. In *Parallel Problem Solving from Nature–PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings 13*. Springer, 211–221.

[39] Ryan J. Urbanowicz, Delaney Granizo-Mackenzie, and Jason H Moore. 2012. Using expert knowledge to guide covering and mutation in a michigan style learning classifier system to detect epistasis and heterogeneity. In *Parallel Problem Solving from Nature-PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I 12*. Springer, 266–275.

[40] Ryan J Urbanowicz, Jeff Kiralis, Nicholas A Sinnott-Armstrong, Tamra Heberling, Jonathan M Fisher, and Jason H Moore. 2012. GAMETES: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData Mining* 5, 1 (2012), 1–14.

[41] Ryan J. Urbanowicz, Marc Meeker, William La Cava, Randal S. Olson, and Jason H. Moore. 2018. Benchmarking relief-based feature selection methods for bioinformatics data mining. *Journal of Biomedical Informatics* 85 (2018), 168–188.

[42] Ryan J Urbanowicz and Jason H Moore. 2015. ExSTraCS 2.0: description and evaluation of a scalable learning classifier system. *Evolutionary Intelligence* 8, 2 (2015), 89–116.

[43] Ryan J Urbanowicz, Randal S Olson, and Jason H Moore. 2016. Pareto inspired multi-objective rule fitness for noise-adaptive rule-based machine learning. In *International Conference on Parallel Problem Solving from Nature*. Springer, 514–524.

[44] Ryan J Urbanowicz, Randal S Olson, Peter Schmitt, Melissa Meeker, and Jason H Moore. 2018. Benchmarking relief-based feature selection methods for bioinformatics data mining. *Journal of biomedical informatics* 85 (2018), 168–188.

[45] Ryan J. Urbanowicz, Robert Zhang, Yuhan Cui, and Pranshu Suri. 2023. STREAMLINE: a simple, transparent, end-to-end automated machine learning pipeline facilitating data analysis and algorithm comparison. In *Genetic Programming Theory and Practice XIX*. Springer, 201–231.

[46] Matthew J Vowels. 2024. Trying to outrun causality with machine learning: Limitations of model explainability techniques for exploratory research. *Psychological Methods* (2024).

[47] Stewart W Wilson. 1995. Classifier fitness based on accuracy. *Evolutionary computation* 3, 2 (1995), 149–175.

[48] Alexa Woodward, Harsh Bandhey, Jason H Moore, and Ryan J. Urbanowicz. 2024. Survival-LCS: A Rule-Based Machine Learning Approach to Survival Analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 431–439.

[49] Robert Zhang, Rachael Stolzenberg-Solomon, Shannon M Lynch, and Ryan J Urbanowicz. 2021. LCS-DIVE: An Automated Rule-based Machine Learning Visualization Pipeline for Characterizing Complex Associations in Classification. *arXiv preprint arXiv:2104.12844* (2021).

[50] Robert Zhang and Ryan J. Urbanowicz. 2022. scikit-ExSTraCS. https://github.com/UrbsLab/scikit-ExSTraCS.

Received 29 January 2025; accepted 19 March 2025