Rule Induction Using Set-Based Particle Swarm Optimisation

Jean-Pierre van Zyl
Division of Computer Science
Stellenbosch University
Stellenbosch, South Africa
20706413@sun.ac.za

Andries P. Engelbrecht
Division of Computer Science, Department of Industrial Engineering
Stellenbosch University
Stellenbosch, South Africa
engel@sun.ac.za

***Abstract*—This paper presents a new approach to induce a list of rules from a dataset by using a set-based particle swarm optimisation algorithm. Many contemporary rule induction algorithms tend to use similar information gain based approaches to fit a training dataset. The proposed novel algorithm is a meta-heuristic approach which finds an optimal rule list while providing the flexibility to overcome traditional drawbacks such as overfitting and rigidity to the datatypes that can be used. This paper shows that the proposed algorithm performs comparatively well when compared to existing rule induction algorithms and it has the potential to be expanded further by adding rule pruning techniques.**

***Index Terms*—rule induction, particle swarm optimisation, set-based particle swarm optimisation**

I. Introduction

Classification problems are a class of supervised learning problems in which models are induced to have the ability to separate unseen data points into a distinct class.

Various approaches to solving classification problems exists: from statistic-based algorithms such as the naïve Bayes classifier [1] to tree induction algorithms such CART [2] and the information gain technique used in ID3 [3]. Nature-inspired approaches to classification have also been explored. For example, highly accurate models have been induced by neural networks (NNs) [4]. The drawback of standard classification solvers, is that it is often difficult to understand why a model classified an instance into the class it did. The severity of this drawback varies from approach to approach, with decision trees offering limited interpretability from the tree structure, while NNs are complete *black boxes*, meaning that additional rule extraction needs to be performed to discern information from the model other than the final classification [5].

As an expansion of general classification abilities, rule induction (RI) algorithms were created and have since been applied to a variety of problems [6]. Rule induction algorithms give the end user an explanation of how a classification was made by returning a set of rules outlining the conditions which need to be satisfied in order for an instance to be classified as a specific class. This paper proposes a new approach to rule induction from a classification dataset, one which uses a set-based particle swarm optimisation (SBPSO) algorithm [7] in a separate-and-conquer approach to find an optimal set of rules. In order to apply the SBPSO algorithm, rule induction is first defined as a set-based discrete optimisation problem. The SBPSO rule induction algorithm is evaluated on a number of classification datasets, and its performance is compared to popular rule induction algorithms. The use of a SBPSO to induce rule provides advantages over classical approaches, as this paper shows. A SBPSO can solve a wider range of problem types by changing the universe generation function, it is not limited to traditional greedy performance metrics, and it holds promise for further expansion for use in dynamic environments where concept drift is present, as well as for multi-objective problems. The results show that this set-based approach performs adequately on the tested datasets and obtains results comparable to the current state of the art algorithms. To the knowledge of the authors, this is the first attempt to apply a set-based optimisation algorithm to rule induction.

The remaining sections of this paper are organised as follows: Section II contains a literature review on relevant works used in the implementation of this new approach or works necessary for the comparison and evaluation of its performance. Section III outlines the proposed rule induction approach and how it was implemented using a SBPSO, while Section IV explains the steps taken to evaluate the performance of this paper's approach. Finally, Section V presents the results on the benchmark problems and Section VI concludes this paper.

II. Background

This section provides a general overview of rule induction concepts and some existing algorithms. The algorithms will be used for performance comparisons in Section V.

A. Rule Induction

In the world of big data, it becomes more difficult to find useful information among the immense amount of new data created by modern systems each second. Therefore, knowledge discovery from databases (KDD) has become an increasingly important aspect of data mining and artificial intelligence systems. Rule induction algorithms give the end user the ability to explain the structure of the decision making process and

978-1-6654-6708-7/22/$31.00 ©2022 IEEE

the relationship between the target and feature variables. The ability to justify a decision is very valuable in certain industries where there are strict regulations or severe consequences for a responsible human agent. This includes the medical field, where doctors cannot afford to make blind decisions, or credit agencies, where service providers must justify the decisions they make.

Generally, rules can be induced and formatted to a human-readable *IF-THEN* format. The first part of the rule, following the IF, is called the antecedent and is used to describe instances. A rule covers an instance if all selectors in the antecedent of the rule are true for the relevant attribute values of the instance. The second part of the rule, following the THEN, is the consequent. When an instance is covered by a rule, that instance is assigned the value of the target class in the consequent. If the rule does not cover the instance, the instance is tested against the antecedent of the next rule in the rule list. Although there are more complex representations of the consequent, for the purpose of this paper the consequent is a value of the target class.

A popular mechanism used in rule induction algorithms is the set-covering also referred to as a separate-and-conquer, approach [8]. The basic premise of set-covering is to find a rule which, according to a predefined metric, best describes the current dataset. Then, the instances in the dataset which have been covered by the induced rule are removed, and the process is repeated to induce the next rule in the set. A basic set-covering approach is presented in Algorithm 1.

Algorithm 1 Basic set-covering approach

```
Define input dataset as E
Initialise rule set R = ∅
while E contains instances do
    Induce new rule r
    Remove all instances from E covered by r
    Add r to R
end while
Return R
```

B. PRISM

PRISM is a rule induction algorithm proposed by Cendrowska [9] which aims to induce less complex rules from a dataset compared to the rules extracted by classification tree algorithms. In contrast to tree-based algorithms which require a common attribute between rules to form a root node, PRISM can split the dataset on any of the input variables. According to Cendrowska, this allows for less complex rule sets where rules have no common attributes when compared to tree-based algorithm which would have to make multiple dataset splits to achieve the same information gain [9].

By maximizing the information gained by each new rule, the entropy for the resulting dataset is minimised and PRISM can induce a rule set which partitions the training dataset into homogenous groups. This strategy achieves 100% classification accuracy on the training dataset, which poses a problem for the generalisation capabilities of PRISM as the rules tend to overfit.

C. C4.5

The C4.5 algorithm was developed by Quinlan as an improvement over the ID3 algorithm [10]. C4.5 is used to induce classification trees by creating the optimal tree branch splits to specialise rules. These optimal splits are chosen by maximising the normalised information gain metric when choosing new selectors to add to the antecedent. The information gain maximisation process is followed recursively until a base case is encountered, after which a new decision boundary is created. The resultant tree can be used as a basis from which to extract rules, or can be used as a standalone classification system [11], [12].

D. RIPPER

Traditional approaches to rule pruning would induce a full rule set and retrospectively use a pruning dataset to prune rules. Fürnkranz and Widmer improved rule pruning by proposing incremental reduced error pruning (IREP) which interleaves the pruning process with the inducing process in order to improve performance [13]. Repeated incremental pruning to produce error reduction (RIPPER) is a RI algorithm proposed by Cohen [14] which was developed as a further improvement over IREP. Cohen aimed to combine the efficiency of IREP and the classification power of C4.5 to develop an approach which would perform well on large, noisy datasets. RIPPER achieves this by essentially applying IREP multiple times to refine the set of induced rules by increasing their accuracy.

E. PART

PART, referring to the "partial decision trees" used in its rule induction process, was developed by Frank and Witten to induce rules from a dataset without the need for a global optimisation process [15]. In order to induce a rule, PART creates a small decision tree from the training dataset by expanding tree subnodes until a stable subtree has been found that cannot be simplified further. The leaf node of the tree that satisfies the highest number of instances is then used to extract a rule, and the covered instances are removed before repeating the process. Frank and Witten believe this approach avoids "hasty generalisation" that results in over-pruning the newly induced rules.

F. Set-Based Particle Swarm Optimisation

Particle swarm optimisation (PSO) is an optimisation algorithm developed by Eberhardt and Kennedy to solve real-valued optimisation problems in a d-dimensional problem space [16]. The PSO algorithm works by mimicking the flight behaviour patterns of birds, with each particle *moving* through the search landscape in search of an optimal solution. The positions of the particles are updated based on their velocity; the velocity is influenced by its previous velocity, an attraction factor to the previously found best position of the particle, and the best position found by any particle in the neighbourhood.

Langeveld and Engelbrecht developed an extension of the PSO algorithm, the set-based particle swarm optimisation (SBPSO) algorithm, to function in discrete environments where the search landscape is generated by a universal set and where solutions are represented as sets [7]. This SBPSO algorithm was developed to solve the multi-dimensional knapsack problem and has since been applied to solve various problems including feature selection, portfolio optimisation and polynomial induction [17]–[19]. The SBPSO algorithm uses analogies of the equations from the original PSO to add or remove elements from the position sets of the particles in its swarm. The velocity update equation of the SBPSO includes factors like the attraction to the previously found best position by a particle, the attraction to the best position found by any neighbourhood of particles, an element removal operator, and an element addition operator (the way these factors are combined is shown in Equation 8). Algorithm 2 outlines the pseudocode of the SBPSO algorithm.

Algorithm 2 Set-Based Particle Swarm Optimization

```
Generate the universal set
Create a swarm containing n_s particles
Initialize particle positions as random subsets of U
Initialize local and global best values
while Stopping conditions not true do
  for each particle i = 1, ..., n_s do
    Evaluate the fitness of the position of particle i.
    if f(X_i) < f(Y_i) then
      Update local best: Y_i = X_i
    end if
    if f(Y_i) < f(Ŷ_i) then
      Update global best: Ŷ_i = Y_i
    end if
  end for
  for each particle i = 1, ..., n_s do
    Update particle i's velocity using equation 8
    Update particle i's position using equation 9
  end for
end while
```

In order to define the necessary SBPSO operators, let $\mathcal{P}(U)$ denote the power set of the universal set (meaning the set of all possible subsets) and $A \times B$ denote the Cartesian product of two sets A and B.

The addition of two velocities, $V_1 \oplus V_2$, is a mapping, $\oplus : \mathcal{P}(\{+,-\} \times U)^2 \rightarrow \mathcal{P}(\{+,-\} \times U)$, that takes two velocities and yields one velocity. Implemented as set operations, a velocity added to a velocity is interpreted as the union operator:

$$V_1 \oplus V_2 = V_1 \cup V_2 \tag{1}$$

The difference between two positions, $X_1 \ominus X_2$, is a mapping, $\ominus : \mathcal{P}(U)^2 \rightarrow \mathcal{P}(\{+,-\} \times U)$, that takes two positions and yields a velocity. The result is effectively the set operation steps which are required to convert X_2 into X_1:

$$X_1 \ominus X_2 = (\{+\} \times (X_1 \backslash X_2)) \cup (\{-\} \times (X_2 \backslash X_1)) \tag{2}$$

The scalar multiplication of a velocity, $\eta \otimes V$, is a mapping, $\otimes : [0,1] \times \mathcal{P}(\{+,-\} \times U) \rightarrow \mathcal{P}(\{+,-\} \times U)$, which takes a scalar and a velocity, and yields a velocity. The mapping results in a randomly selected subset of size $\lfloor \eta \times |V| \rfloor$ from V. Note that $0 \otimes V = \emptyset$ and $1 \otimes V = V$.

$$\eta \otimes V = \text{random subset}(V, \eta) \tag{3}$$

The addition of a velocity and a position, $X \boxplus V$, is a mapping, $\boxplus : \mathcal{P}(U) \times \mathcal{P}(\{+,-\} \times U) \rightarrow \mathcal{P}(U)$, that takes a position and velocity and yields the resultant position.

$$X \boxplus V = V(X) \tag{4}$$

which involves applying the operation associated with each v_i from $V = \{v_1, \ldots, v_n\}$ to X by adding or removing each e_i as dictated by the elements in the velocity.

The removal of elements, $\beta \odot^- S$ from a position $X(t)$, where S is shorthand for $X(t) \cap Y(t) \cap \hat{Y}(t)$, is the mapping, $\odot^- : [0, |S|] \times \mathcal{P}(U) \rightarrow \mathcal{P}(\{+,-\} \times U)$, that takes a scalar and a set of elements, and yields a velocity. The operator is implemented by randomly selecting a subset of elements from S, with a size determined by β, to be removed from $X(t)$:

$$\beta \odot^- S = \{-\} \times \left(\frac{N_{\beta,S}}{|S|} \otimes S \right) \tag{5}$$

The number of elements selected, $N_{\beta,S}$ is defined as:

$$N_{\beta,S} = \min \left\{ |S|, \lfloor \beta \rfloor + \mathbf{1}_{\{r < \beta - \lfloor \beta \rfloor\}} \right\} \tag{6}$$

for a random number $r \sim U(0,1)$; $\mathbf{1}_{\{\text{bool}\}}$ is 1 if bool is true and 0 if bool is false.

The addition of elements, $\beta \odot^+ A$ to a position $X(t)$ where A is shorthand for $U \backslash (X(t) \cup Y(t) \cup \hat{Y}(t))$, is a mapping $\odot^+ : [0, |A|] \times \mathcal{P}(U) \rightarrow \mathcal{P}(\{+,-\} \times U)$ that takes a scalar and a set of elements and yields a velocity. The operator is implemented by randomly selecting a subset of elements from A, with a size determined by β, to be added to $X(t)$:

$$\beta \odot^+ A = \{+\} \times k\text{-Tournament Selection}(A, N_{\beta,A}) \tag{7}$$

where $N_{\beta,A}$ is the number of elements to be added to $X(t)$ as defined in equation (6) and k is a user-defined parameter.

The velocity update equation is defined as:

$$\begin{aligned} V_i(t+1) = {} & c_1 r_1 \otimes (Y_i(t) \ominus X_i(t)) \oplus c_2 r_2 \\ & \otimes \left(\hat{Y}_i(t) \ominus X_i(t) \right) \oplus \left(c_3 r_3 \odot^+ A_i(t) \right) \\ & \oplus \left(c_4 r_4 \odot^- S_i(t) \right) \end{aligned} \tag{8}$$

where $A_i(t)$ and $S_i(t)$ are calculated independently for each particle; $c_1, c_2 \in [0,1]$ and $c_3, c_4 \in [0, |U|]$; each r_k is independently drawn from the distribution $U(0,1)$.

Finally, after the velocity set has been calculated using the operators and equations defined above, the position update equation is defined using the velocity set as:

$$X_i(t+1) = X_i(t) \boxplus V_i(t+1) \tag{9}$$

The equations in this section were developed by the original authors of the SBPSO and are implemented in this paper to induce rules, however, the removal operator $\odot^-$ was modified to implement tournament selection in the same fashion as the $\odot^+$ operator.

III. Inducing Rules with a Set-Based Particle Swarm Optimiser

The following section outlines how the new RI algorithm works as well as the motivation for this new approach.

A. Using a Set-Based Particle Swarm Optimiser as a Separate-and-Conquer Algorithm

In order to use a SBPSO as part of a set-covering algorithm, a protocol needs to be defined for how instances are going to be targeted and when they are going to be removed. To achieve this, each independent instance of a SBPSO simulation is used to induce one rule. The target class for which the SBPSO induces rule is selected as the majority class of the remaining instances in the dataset. This is contrary to existing algorithms, which tend to start with the minority class, because using a meta-heuristic does not strive for complete homogeneity in the instances it covers. The SBPSO is able to compromise while balancing the contributing factors in the objective function, and this compromise can lead to poor performance when inducing rules for the minority class. The SBPSO then runs for a predetermined number of iterations and the resulting global best position is adopted as the new rule. This rule is then added to the rule list and is used to remove the covered instances in the dataset. The process repeats to induce for the current majority class until no more instances are available for the majority class. The next majority class is then determined and rules are induced for that majority class. This process terminates once all instances in the dataset have been covered by a rule.

B. Motivation for a meta-heuristic approach

Rule induction using a SBPSO (RiSBPSO) can be achieved on a dataset by using the particles in its swarm to search the fitness landscape for possible combinations of selectors to make up an optimal rule antecedent. This approach is contrary to popular *gain-based* approaches where rules are built iteratively by building an antecedent either through a top-down or bottom-up approach. Instead, the exploration-exploitation trade-off mechanisms intrinsic to the velocity update equation of the SBPSO are used to search and refine for complete rules out of all possible selector combinations as candidate antecedents. In order to induce rules for a dataset using a SBPSO the following algorithm features need to be designed: (1) an appropriate selector representation, (2) a universe generator, and (3) an objective function.

Most of the classical rule induction algorithms discussed in Section II tend to be greedy algorithms which overfit the training dataset. This section outlines how the SBPSO meta-heuristic can be applied to solve the same rule induction problems while avoiding overfitting and theoretically reducing the need to prune the rule set. Using a SBPSO mitigates the performance issues associated with the curse of dimensionality by allowing variable particle position and velocity sizes, in other words the positions sets of the particles do not have to contain all feature variables. When compared to classical RI approaches, RiSBPSO also allows more explorative freedom on the selectors in the antecedents of its rules. This leaves room for modified comparison operators to be used meaning that the RiSBPSO algorithm can be adapted to be used on different attribute types or be extended from propositional comparisons to first-order logic. Algorithm 3 provides the steps of the RiSBPSO.

Algorithm 3 RiSBPSO

Let E be the input dataset
Let R be the initial empty rule set
for Select the majority class as the target class, C_i **do**
  Induce a SBPSO swarm with a target variable C_i
  Let the resulting global best position be r
  Add r to R
  Remove the instances covered by the r from the E
end for

C. Universe and Selector Representation

The universal set functions as the landscape representation for the problem being solved by a SBPSO. While the number of independent variables (*i.e.* descriptive features) in the dataset may be n_x, the dimensionality of the problem being solved in a SBPSO depends on the cardinality of the universal set, $|U|$. However, dimensions of the particles are not fixed at $|U|$, but are in the range $[1, |U|]$. This variable length position size aids in breaking the curse of dimensionality, because position sizes can remain relatively small even when the size of the universal set increases.

Each element in the universe is a tuple which consists of an attribute, an operator and a value. The attributes are the independent input variables of the dataset, and each attribute has its own set of possible compatible values. Compatible values are either other feature variables (when dealing with first-order logic) or constants that occur in the relevant attribute column. Further, the chosen operators dictate which comparisons can be made between the attributes and their possible values. In this proposed algorithm, the equality operator and its negation are used (*i.e.*$=, \neq$).

In RiSBPSO, the antecedents contain one or more selectors which are used to specialise a rule and to determine which data instances are covered by the rule. The selectors are in the form $(a_i, \langle operator \rangle, v_j)$, where a_i is one of the input features and v_j is a compatible constant from the list of possible values of the attribute a_i. These selectors are then combined conjunctively to specialise the rule. An antecedent, A, of length l, is formed as a conjunction of the selectors s, *i.e.* $A = s_0 \wedge s_1 \wedge \cdots \wedge s_{l-1}$.

Given the hypothetical dataset D with characteristics as set out in Table I, the generated universe U is shown in Equation (10):

TABLE I
SUMMARY OF LEGITIMATE ATTRIBUTE VALUES IN A HYPOTHETICAL DATASET

Attribute	Possible Values
a_1	$\{v_{1,1}, v_{1,2}\}$
a_2	$\{v_{2,1}, v_{2,2}, v_{2,3}\}$
a_3	$\{v_{3,1}, v_{3,2}\}$

$$
\begin{aligned}
U = \\
&\{(a_1, =, v_{1,1}), (a_1, \neq, v_{1,1}), (a_1, =, v_{1,2}), (a_1, \neq, v_{1,2}), \\
&(a_2, =, v_{2,1}), (a_2, \neq, v_{2,1}), (a_2, =, v_{2,2}), (a_2, \neq, v_{2,2}), \\
&(a_2, =, v_{2,3}), (a_2, \neq, v_{2,3}), (a_3, =, v_{3,1}), (a_3, \neq, v_{3,1}), \\
&(a_3, =, v_{3,2}), (a_3, \neq, v_{3,2})\}
\end{aligned} \tag{10}
$$

Generally, the size of the universal set is calculated as

$$|U| = |O| \cdot \sum_{a=1}^{j} |V_a| \tag{11}$$

where O is the set containing the operators, V_a is the set of legitimate values for attribute a_a, and j is the number of attributes in the dataset.

D. Particle Representation and Objective Function

Each particle in the swarm is defined by four sets: a position set, a personal best position set, a neighbourhood best position set, and a velocity set. The three position sets are all subsets of the powerset of the universe ($\mathcal{P}(U)$), while the velocity set is a subset of $\mathcal{P} \times \{+, -\}$ (the Cartesian product of the powerset of U with $\{+, -\}$). When applying an induced antecedent (represented as a set of selectors) to a dataset, each selector is conjunctively used to specialise the subset of data which is deemed *covered* by the rule.

The key to applying a meta-heuristic like the SBPSO to different problems successfully lies in the definition of the objective function. The objective function quantifies the aptness of a candidate solution during the search for a global optimum.

Rule induction is essentially a multi-objective optimisation problem which requires that the contributions from each of the sub-objectives in the fitness function be combined in an appropriate way. This is achieved by applying weight-aggregation in the objective function by using the coefficients of each sub-objective in the fitness function to control the trade-off between the often conflicting goals.

For RI, the sub-objectives are to

1) maximise the coverage of the rule,
2) maximise the accuracy of the rule,
3) maximise the purity of the rule,
4) minimise the complexity of the rule, and
5) minimise the entropy of the labels of the covered instances.

For a dataset $D = \{X, \boldsymbol{y}\}$ with I instances of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I$ with corresponding labels $y_1, \ldots, y_I$ and a rule R consisting of an antecedent A and consequent C, the following is defined: an instance $\boldsymbol{x}_p = (x_1, \ldots, x_d)$ is considered covered by a rule antecedent if each selector $s_k \in A$ evaluates as true. Further, $\boldsymbol{x}_i$ is considered positively covered if it is covered and $y_i = C$; otherwise, if it is covered and $y_i \neq C$, then the instance is negatively covered. Table II defines the necessary quantities used to calculate the fitness of a particle.

TABLE II
TABLE OF QUANTITIES USED IN METRIC CALCULATIONS

Symbol	Description
r	The induced rule being evaluated
P	The number of instances in the data set which are of the target class
p	The number of positively covered instances by r. Note that $0 \leq p \leq P$
N	The number of instances in the data set which are not of the target class
n	The number of incorrectly covered instances by r. Note that $0 \leq n \leq N$
T	The total number of instances which need to be covered. Note that $P + N = T$
t	The number of instances which are (correctly or incorrectly) covered by r. Note that $p + n = t$
l	The number of selectors in A
e	The entropy of the labels in the set of covered instances

The objective function is defined as

$$
\begin{aligned}
f(D, r) = w_1 \left(1 - \mathcal{A}(D, r)\right) + w_2 \left(1 - \mathcal{P}(D, r)\right) \\
+ w_3 \left(1 - \mathcal{L}(D, r)\right) + w_4 \left(\mathcal{S}(D, r)\right) + w_5 \left(\mathcal{E}(D, r)\right)
\end{aligned} \tag{12}
$$

where

- $\mathcal{A}(D, r) = \left(\frac{p+(N-n)}{P+N}\right)$ refers to the accuracy of the rule,
- $\mathcal{P}(D, r) = \left(\frac{p}{p+n}\right)$ refers to the purity of the rule,
- $\mathcal{L}(D, r) = \left(\frac{p+1}{p+n+2}\right)$ refers to the Laplace estimator,
- $\mathcal{S}(D, r) = \left(\frac{l}{|U|}\right)$ refers to the size of the rule, and
- $\mathcal{E}(D, r) = e$ refers to the entropy of the covered instances.

Each $w_i \in [0, 1]$ and $\sum_{i=1}^{5} w_i = 1$. This objective is treated as a minimisation problem, but instead of merely taking the dual of the two maximisation objectives, the problem was formulated that each sub-objective will fall into the range $[0, 1]$. This combined with the restrictions on the objectives' weights ensures that the results are more interpretable as the final value will always be $[0, 1]$.

IV. EMPIRICAL PROCESS

This section outlines the processes followed to evaluate the proposed RiSBPSO algorithm and to compare it to existing rule induction algorithms.

A. Benchmark Problems

In order to evaluate the feasibility of this RI approach, various open-access datasets popular in existing literature were tested. These benchmark problems were obtained from the UCI machine learning repository. Each problem set is a classification problem, with only categorical features to allow

consistent comparison between the classification performance of each RI algorithm and to simplify the universal set generation process.

Minimal preprocessing was done to the datasets. Attributes with I unique values were removed, as well as attributes with only one unique value because they are irrelevant. Instances with missing values were discarded, unless in the cases where a 10% these of missing values were of the same attribute then the attribute was removed instead of the individual instances. Table III describes the number of attributes and instances for each benchmark dataset after preprocessing.

TABLE III
BENCHMARK DATASETS CHARACTERISTICS

Dataset	Number of input attributes	Number of instances
Audiology	68	216
Balance Scales	4	625
Breast Cancer	9	277
Car	6	1728
Congressional Voting	16	435
Hayes-Roth	4	132
KR vs KP	36	3196
Lymphography	17	148
Monks-1	6	556
Monks-2	6	601
Monks-3	6	553
Mushroom	21	8124
Nursery	8	12959
Primary Tumour	15	335
SPECT	21	267
Tic-tac-toe	9	957
Zoo	16	101

B. Algorithm Tuning

The algorithms were tuned by sampling possible control parameter combinations from a quasi-randomly generated set of Sobol sequences in order to maximise performance [20]. These sequences are generated by a predetermined formula to provide good coverage of the hypercube generated by the control parameter ranges, while adding more stochasticity to the values when compared with a grid search. The parameters were tuned independently for each dataset, with each tuning process sampling 128 Sobol sequences and using 30 swarm particles running for 300 iterations. The optimal combination of parameters for each dataset was found by tuning on a training set of the dataset, and using the generalisation performance to select the best combination. This optimal combination of parameters was then used to perform the final tests for the given dataset.

For the comparison algorithms that have tunable parameters, 128 Sobol sequences were also generated to tune their control parameters. Table IV contains the list of parameters and their ranges tuned for each algorithm, and Table V contains the values for the found optimal parameter values (for brevity, values have been rounded to 3 decimal places).

C. Performance Measures

For each of the benchmark problems, the accuracy over the test set is reported. The datasets were split randomly into a training set, consisting of 70% of the instances, and a test set of the remaining 30%. For datasets which have a dedicated test set (like the Monks problems), this independent set was incorporated into the main dataset to ensure that the test sets in each independent run were sampled randomly.

The final tests were performed with the optimally selected control parameters, for 300 iterations with 30 particles. Each of these tests was repeated for 30 independent tests to ensure statistically relevant results.

TABLE IV
CONTROL PARAMETERS TUNED PER ALGORITHM

SBPSO		C4.5	
Parameter	**range**	**Parameter**	**range**
c_1	$[0, 1]$	$conf$	$(0, 1)$
c_2	$[0, 1]$	min	$\{1 \dots 10\}$
c_3	$[0.5, 5]$		
c_4	$[0.5, 5]$		
RIPPER		**PART**	
Parameter	**range**	**Parameter**	**range**
k	$\{1 \dots 5\}$	$conf$	$(0, 1)$
		min	$\{1 \dots 10\}$

V. RESULTS

The following section presents the results obtained after apply each respective algorithm to the datasets discussed in Subsection IV-A.

It is clear from Table VI that the RiSBPSO performs well on the datasets and is able to achieve consistent results, as shown by the small standard deviations. On most datasets, the mean accuracies on the test set of the algorithms are within one standard deviation of each other. There are certain exceptions, like the Monks-2 dataset, where RiSBPSO consistently outperforms the other algorithms by about 20%. RiSBPSO was outperformed by a 10% margin on the Audiology and SPECT datasets by C4.5.

Table VII breaks down the results for each RI algorithm. The average accuracy for RiSBPSO was $\mathbf{83.31 \pm 2.80}$, for PRISM it was 75.49 ± 2.82, RIPPER achieved 80.90 ± 3.33, while PART's average was 83.09 ± 2.92 and C4.5's was 81.84 ± 3.33. RiSBPSO and C4.5 both had the best performance for 6 datasets, compared to 4, 1 and 0 for PART, RIPPER and PRISM respectively. However, the better average test set accuracy and lower standard deviation indicates that RiSBPSO has both a smaller bias and is more consistent in its performance. This shows that RiSBPSO is not only able to compete with the state of the art algorithms but can also marginally out perform them.

VI. CONCLUSION

The purpose of this paper was to present a new approach to rule induction and to compare the proposed approach with traditional rule induction algorithms. The RiSBPSO algorithm performs comparatively on par with the current state of the art RI algorithms and on average outperformed all the comparison algorithms. The prospects of the RiSBPSO are further improved by the fact the compared algorithms implement pruning

TABLE V
OPTIMAL CONTROL PARAMETERS FOR IMPLEMENTED RI ALGORITHMS

Dataset	SBPSO c_1, c_2, c_3, c_4	C4.5 $conf$, min
Audiology	$0.512, 0.881, 3.623, 1.465$	$0.385, 1$
Balance Scale	$0.995, 0.897, 2.433, 4.919$	$0.897, 2$
Breast Cancer	$0.038, 0.624, 4.483, 0.821$	$0.470, 6$
Car	$0.454, 0.793, 2.769, 1.209$	$0.946, 2$
Congressional Voting	$0.990, 0.556, 4.271, 2.708$	$0.378, 2$
Hayes-Roth	$0.950, 0.123, 2.688, 3.989$	$0.772, 4$
KR vs KP	$0.896, 0.833, 2.392, 2.562$	$0.168, 1$
Lymphography	$0.075, 0.402, 2.132, 3.807$	$0.851, 9$
Monks-1	$0.223, 0.407, 1.566, 1.081$	$0.171, 1$
Monks-2	$0.390, 0.652, 3.053, 1.966$	$0.918, 1$
Monks-3	$0.894, 0.059, 1.905, 0.998$	$0.348, 7$
Mushroom	$0.651, 0.468, 3.024, 3.652$	$0.339, 4$
Nursery	$0.628, 0.596, 4.757, 4.012$	$0.630, 1$
Primary Tumour	$0.673, 0.551, 4.691, 2.016$	$0.001, 3$
SPECT	$0.122, 0.367, 4.945, 4.796$	$0.352, 4$
Tic-Tac-Toe	$0.837, 0.638, 3.441, 3.097$	$0.816, 2$
Zoo	$0.100, 0.988, 4.369, 1.475$	$0.714, 1$

Dataset	RIPPER k	PART $conf$, min
Audiology	1	$0.655, 1$
Balance Scale	5	$0.916, 8$
Breast Cancer	3	$0.789, 10$
Car	4	$0.622, 1$
Congressional Voting	1	$0.490, 3$
Hayes-Roth	3	$0.723, 5$
KR vs KP	4	$0.825, 1$
Lymphography	4	$0.390, 1$
Monks-1	2	$0.695, 4$
Monks-2	2	$0.917, 5$
Monks-3	1	$0.312, 5$
Mushroom	1	$0.363, 6$
Nursery	1	$0.311, 2$
Primary Tumour	1	$0.420, 6$
SPECT	2	$0.706, 9$
Tic-Tac-Toe	2	$0.925, 1$
Zoo	2	$0.956, 1$

techniques, while RiSBPSO can still be extended to implement pruning.

This paper showed that a set-based approach to RI is possible, and that the SBPSO algorithm is suited to be used for this purpose. The authors believe that through further study of the factors influencing RiSBPSO's performance, even better results can be obtained. These factors include the elemental mechanisms built into the SBPSO algorithm, as well as the subcomponents in the objective function. These objective function subcomponents can be modified and tested, and the user-defined weight coefficients can be tuned in future studies. Additionally, it can be studied how well RiSBPSO performs when rule pruning is applied.

TABLE VI
RI PERFORMANCE METRICS FOR RISBPSO

Dataset	Accuracy	Number of rules	Rule size
Audiology	68.77 ± 5.52	43.63 ± 6.25	1.96 ± 0.19
Balance Scales	77.32 ± 2.68	47.9 ± 3.33	3.22 ± 0.12
Breast Cancer	65.76 ± 4.43	33.47 ± 2.9	6.06 ± 0.54
Car	96.07 ± 1.07	42.9 ± 2.93	2.99 ± 0.11
Congressional Voting	93.67 ± 2.1	9.87 ± 1.45	2.43 ± 0.43
Hayes-Roth	70.83 ± 5.26	17.7 ± 1.44	2.42 ± 0.25
KR vs KP	99.13 ± 0.35	22.8 ± 2.27	2.53 ± 0.23
Lymphography	38.07 ± 7.65	24.3 ± 1.59	4.54 ± 0.36
Monks-1	100.0 ± 0.0	6.9 ± 0.7	1.83 ± 0.24
Monks-2	97.46 ± 2.15	30.67 ± 2.07	2.8 ± 0.16
Monks-3	98.96 ± 0.64	4.47 ± 1.18	1.36 ± 0.18
Mushroom	100.0 ± 0.0	4.23 ± 0.5	3.62 ± 0.39
Nursery	99.17 ± 0.27	141.9 ± 10.1	4.34 ± 0.09
Primary Tumour	33.53 ± 4.13	99.0 ± 6.2	4.04 ± 0.15
SPECT	84.2 ± 5.14	21.03 ± 7.03	3.52 ± 0.78
Tic-tac-toe	98.44 ± 1.83	17.0 ± 3.11	2.75 ± 0.17
Zoo	95.05 ± 4.46	8.07 ± 0.77	1.2 ± 0.1

ACKNOWLEDGEMENTS

The authors thank the National Research Foundation (NRF) for providing funding and the Centre for High Performance Computing (CHPC) for providing computational resources.

REFERENCES

[1] H. Zhang, "The optimality of naive bayes," in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, vol. 2, 01 2004.

[2] W. Buntine, "Learning classification trees," *Statistics and Computing*, vol. 2, pp. 63–73, 1992.

[3] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, pp. 81–106, 1986.

[4] E. A. Wan, "Neural network classification: A bayesian interpretation," *IEEE Transactions on Neural Networks*, vol. 1, no. 4, pp. 303–305, 1990.

[5] R. Setiono and H. Liu, "Understanding neural networks via rule extraction," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 480–485.

[6] P. Langley and H. A. Simon, "Applications of machine learning and rule induction," *Communications of the ACM*, vol. 38, no. 11, pp. 54–64, 1995.

[7] J. Langeveld and A. P. Engelbrecht, "Set-based particle swarm optimization applied to the multidimensional knapsack problem," *Swarm Intelligence*, vol. 6, pp. 297–342, 2012.

[8] J. Fürnkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, pp. 3–54, 1999.

[9] J. Cendrowska, "Prism: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, no. 4, pp. 349–370, 1987.

[10] R. Quinlan, "C4.5: Programs for machine learning," *Machine Learning*, vol. 16, 1993.

[11] B. Lakshmi, T. Indumathi, and N. Ravi, "A study on c.5 decision tree classification algorithm for risk predictions during pregnancy," *Procedia Technology*, vol. 24, pp. 1542–1549, 2016.

[12] P. V. Ngoc, C. V. T. Ngoc, T. V. T. Ngoc, and D. N. Duy, "A c4.5 algorithm for english emotional classification," *Evolving Systems*, vol. 10, no. 3, pp. 425–451, 2019.

[13] J. Fürnkranz and G. Widmer, "Incremental reduced error pruning," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 70–77.

[14] W. W. Cohen, "Fast effective rule induction," in *Machine Learning Proceedings*, San Francisco (CA), 1995, pp. 115–123.

[15] E. Frank and I. Witten, "Generating accurate rule sets without global optimization," in *Proceedings of the Fifteenth International Conference on Machine Learning*, 06 1998, pp. 144–151.

TABLE VII
TABLE TO COMPARE RI PERFORMANCE BETWEEN DIFFERENT ALGORITHMS

Dataset	RiSBPSO	PRISM	RIPPER	PART	C4.5
Audiology	68.77 ± 5.52	10.56 ± 2.62	67.58 ± 7.11	75.64 ± 6.29	$\mathbf{78.76 \pm 4.58}$
Balance Scales	$\mathbf{77.32 \pm 2.68}$	61.80 ± 3.30	73.26 ± 2.52	75.29 ± 2.76	65.13 ± 3.42
Breast Cancer	65.76 ± 4.43	63.41 ± 4.60	71.72 ± 3.57	$\mathbf{74.65 \pm 5.79}$	72.77 ± 5.62
Car	96.07 ± 1.07	89.13 ± 1.32	84.53 ± 2.26	$\mathbf{96.15 \pm 0.90}$	92.23 ± 1.30
Congressional Voting	93.67 ± 2.10	93.10 ± 1.82	94.68 ± 1.72	$\mathbf{95.08 \pm 1.60}$	94.58 ± 1.86
Hayes-Roth	70.83 ± 5.26	71.00 ± 5.97	$\mathbf{81.00 \pm 5.06}$	67.58 ± 6.66	69.91 ± 5.57
KR vs KP	99.13 ± 0.35	97.09 ± 0.71	99.07 ± 0.28	99.16 ± 0.31	$\mathbf{99.38 \pm 0.32}$
Lymphography	38.07 ± 7.65	34.39 ± 6.09	41.51 ± 6.47	39.46 ± 6.08	$\mathbf{42.80 \pm 7.21}$
Monks-1	$\mathbf{100.00 \pm 0.00}$	98.84 ± 1.39	87.78 ± 10.24	98.92 ± 2.71	96.06 ± 6.58
Monks-2	$\mathbf{97.46 \pm 2.15}$	73.87 ± 3.56	62.31 ± 3.54	69.51 ± 4.68	68.85 ± 4.73
Monks-3	98.96 ± 0.64	96.04 ± 1.03	98.37 ± 1.00	99.05 ± 0.69	$\mathbf{99.09 \pm 0.59}$
Mushroom	$\mathbf{100.00 \pm 0.00}$	99.99 ± 0.03	99.98 ± 0.03	99.93 ± 0.09	$\mathbf{100.00 \pm 0.00}$
Nursery	$\mathbf{99.17 \pm 0.27}$	97.28 ± 0.39	96.30 ± 0.57	98.86 ± 0.33	98.03 ± 0.29
Primary Tumour	33.53 ± 4.13	28.48 ± 4.06	37.78 ± 4.04	37.72 ± 3.70	$\mathbf{39.57 \pm 5.35}$
SPECT	84.20 ± 5.14	82.79 ± 3.96	92.83 ± 1.93	92.83 ± 2.86	$\mathbf{93.50 \pm 2.61}$
Tic-tac-toe	$\mathbf{98.44 \pm 1.83}$	95.59 ± 1.80	97.71 ± 0.64	96.82 ± 1.65	85.43 ± 2.58
Zoo	95.05 ± 4.46	90.11 ± 5.26	87.33 ± 5.73	$\mathbf{95.88 \pm 2.67}$	95.33 ± 4.00

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks*, 1995, pp. 1942–1948 vol.4.

[17] A. P. Engelbrecht, J. Grobler, and J. Langeveld, "Set based particle swarm optimization for the feature selection problem," in *Engineering Applications of Artificial Intelligence*, vol. 85, 2019, pp. 324–336.

[18] K. Erwin and A. P. Engelbrecht, "Set-based particle swarm optimization for portfolio optimization," in *Proceedings of the Twelfth International Conference on Swarm Intelligence*, 2020, pp. 333–339.

[19] J. van Zyl and A. P. Engelbrecht, "Polynomial Approximation Using Set-Based Particle Swarm Optimization," in *Proceedings of the International Conference on Swarm Intelligence*, ser. Lecture Notes in Computer Science, Y. Tan and Y. Shi, Eds., 2021, vol. 12689, pp. 210–222.

[20] I. M. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 86–112, 1967.