Improving ID3 through Optimized Feature Selection

Samantha Ball
1603701
School of Computer Science and Applied Mathematics
University of the Witwatersrand

***Abstract*—The Iterative Dichotomiser 3 (ID3) classifier is a well-known decision tree algorithm that uses the information gain to select the optimal feature as the splitting condition at each iteration. However, a major drawback of the ID3 is the tendency of the classifier to overfit, therefore inhibiting classification performance on unseen data. In order to overcome this limitation and increase the efficiency of the decision tree structure, feature subset selection can be employed. This introduces the problem of selecting an optimal feature subset which will retain sufficient information to correctly classify data while reducing overfitting. Due to the vast number of potential feature combinations, optimization techniques such as local search or evolutionary techniques can provide a valuable solution. In this investigation we compare the efficacy of the tabu search and genetic algorithm metaheuristics in the context of feature subset selection. We investigate the variants of each algorithm most suitable to the problem and its associated constraints. Through statistical analysis, it is observed that the tabu search approach outperforms the genetic algorithm in the context of feature selection. Furthermore, the feature subsets found from the tabu search exhibit significantly more reliable and advantageous results as an average of 90.5% accuracy is obtained on the test set. This translates to a 2.4% improvement in accuracy over the baseline accuracy obtained from all 100 input features, while simultaneously improving the efficiency of the ID3 classifier.**

***Index Terms*—feature selection, tabu search, genetic algorithm, id3**

I. Introduction

Feature selection presents several key advantages to machine learning classifiers as it works to mitigate against overfitting, reduce redundancy in the input data and minimize prediction time [1]. However, the selection of an appropriate feature subset is a non-trivial task for feature sets with high dimensions. The task of feature selection can thus be modelled as an optimization problem. This investigation seeks to determine the efficacy of the tabu search and genetic algorithm optimization techniques when applied to the problem of feature selection in the context of the ID3 Decision Tree classifier. The ID3 Decision Tree algorithm is a well-known classifier and predecessor to the popular C4.5 algorithm. The ID3 classifier selects features based on the largest possible information gain at each split of the decision tree in a recursive manner. The ID3 algorithm is thus a greedy strategy as it selects the optimal attribute to divide the dataset at each iteration. However, the ID3 classifier is prone to overfitting as it grows until the training data can be classified perfectly, inhibiting its ability to generalise.

In order to improve the generalisation abilities of the ID3 algorithm, feature selection can be used to select an optimal subset of features which constrains the growth of the tree while still retaining sufficient information to classify the input samples. For a relatively large number of features, selecting this optimal subset is time-consuming. Therefore the use of appropriate meta-heuristics can provide a more efficient solution. Consequently, we compare the ability of the tabu search and genetic algorithm metaheuristics to effectively determine the optimal subset of features from an input set of 100 features. The tabu search algorithm provides an example of a local search strategy while the genetic algorithm exhibits the benefits of evolutionary and population-based techniques. Since the computational expense of constructing the ID3 decision tree for every subset is relatively significant, the evaluation of the fitness provides a significant limitation in terms of computational resources. Therefore the efficiency of the meta-heuristic in finding an optimal solution within a reasonable amount of time is crucial.

The tabu search and genetic algorithms approaches are first designed specifically to meet the requirements of the feature selection problem and appropriate control parameters are determined experimentally. The performance of each algorithm is then compared using a Mann-Whitney U test and the classification performance of the final feature subsets is analysed.

The paper is structured as follows. In Section II, we outline the relevant techniques and formulate the feature selection optimization problem. In Section III, we provide an in-depth discussion of the approaches used and the specific design features of the respective metaheuristics. Section IV presents our findings and a comprehensive comparison of the two algorithms. Lastly, Section V summarizes our central findings.

II. Background

The need for appropriate feature selection in the context of the ID3 classifier can be further explored by examining the mechanisms of the ID3 algorithm.

A. The Iterative Dichotimiser 3 Classifier

As outlined above, the Iterative Dichotimiser 3 (ID3) classifier is a greedy, top-down approach to building a decision tree [2]. This implies that the tree is built from the top and selects the best possible feature at each split. The ID3 decision tree uses the information gain to select the best feature. Information gain is defined as the reduction in entropy and therefore measures how well a given feature separates the target classes [2]. Entropy provides a measure of the disorder in the target

features of the dataset. In the case of binary classification with only two target classes, entropy is 0 if all the values in the target column are the same and 1 if the target column has an equal number of values for both target classes. Thus, in the context of classification, entropy is formulated as,

$$H(S) = -\sum_{i=1}^{n} p_i * log_2(p_i) \tag{1}$$

where n is the number of distinct target classes and p_i is the probability of a given target class i. The information gain can then be calculated as follows,

$$IG(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} * H(S_v)) \tag{2}$$

where H denotes the entropy of the set, S is the given dataset and S_v is the set of all rows in the dataset which have value v for feature A.

The ID3 algorithm constructs a decision tree structure in a recursive manner as summarized in the process below.

Algorithm 1 ID3 Decision Tree

Let S denote the set of training samples
Let C denote the set of target classes in the sample data
Initialize root node
if all $s \in S$ have same target class $c \in C$ (pure node) **then**
  Return current node with $label = c$
end if
if $length(featurelist) = 0$ **then**
  Return node with $label = mode(C)$
end if
Find feature F with maximum information gain IG_{max}
Set splitting condition of current node $= F$
for f in F **do**
  Create child node with value = f
  Determine the subset S_f of S with feature $F = f$
  if subset S_f is empty **then**
    Add leaf node with $label = mode(C)$
  else
    Update feature list by removing feature F
    Add the subtree given by recursive function call to $ID3$ with new data subset S_f and updated feature list F
  end if
end for
return root

The algorithm selects the feature with the highest information gain as the splitting condition for the new node until either all target values are the same, the subset of data for a particular child value is empty or no more features are available, resulting in a leaf node with the most probable target value.

B. Feature Selection as an Optimization Problem

Since the ID3 algorithm constructs a decision tree such that no errors are made on the training data, it is prone to overfitting and a corresponding lack of generalisation. Several approaches have been developed to reduce overfitting. These include pruning the decision tree after it has been constructed, such as reduced error pruning or rule post-pruning, or preventing the decision tree from growing to a depth where it perfectly classifies the training data, such as feature subset selection [2].

This paper investigates feature selection as an optimization problem. Accordingly, we consider the search for the best subset of features such that the validation accuracy is maximised, and therefore the feature subset that provides the best generalization ability. We can formulate the feature selection problem as,

$$Select\ S \subseteq F\ such\ that\ A(S)\ is\ maximized \tag{3}$$

where S is our feature subset, F is the set of all features and $A(S)$ is the validation accuracy on the given subset S. Our approach compares two contrasting optimization techniques, tabu search and genetic algorithms.

C. Tabu Search

Tabu search is a classic, iterative search approach which employs a local or neighbourhood search procedure to iteratively move from one potential solution to an improved solution in the neighbourhood. Tabu search provides an extension to the classic hill climbing approach by constructing a tabu list of solutions that have been recently visited and are prohibited from being chosen again, as well as allowing the search to escape local optima by selecting a suboptimal movement. The general procedure for tabu search is outlined below.

Algorithm 2 Tabu Search

Let $f : S \to \mathbb{R}$ be the objective function
Let $\mathcal{N}(x)$ be the neighbourhood operation
Let x be the initial starting position
Let x_{best} be the best solution overall
Let T_{max} denote the max tabu list size
Initialize tabu list $T = \{\}$
while stopping condition false **do**
  $x' \leftarrow argmax_{y \in \mathcal{N}(x)} f(y)$
  if $x' \notin T$ **then**
    $x \leftarrow x'$
  end if
  if $f(x) > f(x_{best})$ **and** $x \notin T$ **then**
    $x_{best} \leftarrow x$
  end if
  Add x to T
  if $length(T) > T_{max}$ **then**
    Remove first element t_0 from T
  end if
end while
return x_{best}

D. Genetic Algorithms

Genetic algorithms are a population-based approach to optimization inspired by Darwinian evolution and form a

subset of the larger class of evolutionary computation. Genetic algorithms draw from the theory of natural selection by incorporating principles such as selection according to survival of the fittest, reproduction through crossover and the introduction of new genetic material through mutation. The core components of selection, crossover and mutation can take many forms and have been an area of active research. An overview of the genetic algorithm approach is given below.

Algorithm 3 Genetic Algorithm

Let $t = 0$ be the generation counter
Initialize an n_x-dimensional population C_0 of population size n_s
while stopping condition false **do**
  Evaluate fitness $f(x_i(t))$ for each individual $x_i(t)$
  Select parents P using the selection operator
  Perform crossover at crossover rate p_c
  Perform mutation at mutation rate p_m
  Select new population C_{t+1}
  $t = t + 1$
end while
$x_{best} = argmax_{x_i \in C(t)} f(x_i)$
return x_{best}

The genetic algorithm therefore returns the individual with the best fitness in the final population. The optimal choice of selection operators and crossover and mutation techniques is problem dependent.

III. Implementation

In order to compare the efficacy of tabu search and genetic algorithms with regard to feature selection for the ID3 classifier, we consider a dataset with 100 features, where each feature can take on the value A or B. The target values are either $True$ or $False$ and thus constitute a binary classification task.

A. ID3 Decision Tree Algorithm

The ID3 decision tree algorithm was implemented as a tree class together with a node class to hold the value of each splitting condition, the feature values A and B for each branch and the corresponding successive child nodes. A recursive function was used to build the tree from the given training data according to the algorithm outlined in the previous section.

A significant optimization was required in order to ensure that the ID3 algorithm could be run in a feasible amount of time. In order for the construction of the tree to be performed within realistic time constraints, the training dataset used to construct the tree was stored as a class variable and then subsequent data subsets for each recurrent tree branch were stored as indices of the corresponding rows. In this way, only a list of indices was required at each recursive step instead of multiple data subsets, thereby minimizing computational expense. This optimization was critical to the practical utilization of the ID3 algorithm in this investigation.

The correct functioning of the ID3 decision tree algorithm was verified by measuring the classification accuracy of the training samples. It was observed that for any number of input features in excess of 50, 100% training accuracy was achieved, thus verifying that the decision tree was correctly constructed and that overfitting was occurring as expected.

B. Tabu Search

The Tabu Search algorithm involves several design alternatives that can be selected to better fit the given optimization problem. Firstly, the way in which the solutions are encoded is pivotal to the design of the algorithm. In order to fulfil our aim of choosing the optimal subset of these features, all possible combinations of features need to be encoded in a compact representation.

B.1. Bitstring encoding

Accordingly, in our tabu seach implementation, each solution is encoded as a bitstring with length equal to the total number of available features. Each bit in the bit string indicates whether that feature is chosen or not. This results in 2^n possible combinations where n is the number of features. In the case of the given data, we have $n = 100$ and thus $2^{100} = 1.27 \times 10^{30}$ possible combinations. Therefore an appropriate search algorithm is necessary to navigate this search space.

B.2. Neighbourhood function

a) One flip neighbourhood

A subsequent design choice affecting the performance of the tabu search algorithm includes the choice of the neighbourhood from which the next candidate solution will be chosen. In order to provide a neighbourhood function consistent with the bitstring representation of the candidate solutions, the one-flip neighbourhood was selected. The one flip neighbourhood consists of all bitstrings within one bit flip of the current candidate solution. Thus for a bitstring with length 100, 100 solutions will lie in the surrounding neighbourhood.

Algorithm 4 One Flip Neighbourhood

Let x represent the current candidate solution
Initialize list N to contain neighbours N_i in neighbourhood $\mathcal{N}(x)$
for i = 0 **to** *length*(x) **do**
  $N_i = x$
  $N_i[i] = not\ x[i]$
  $i = i + 1$
end for
return N

The one flip neighbourhood was chosen due to its complementary nature with the chosen bitstring encoding as well as its ability provide intuitively adjacent solutions in the search

space. Specifically, the one flip neighbourhood represents all subsets of features that are formed by either adding or removing one feature from the original candidate subset. Therefore the concept of closely related neighbourhood solutions is naturally encoded.

However, a drawback of the one flip neighbourhood is the increasingly large neighbourhood size as the number of dimensions of the candidate solutions increases. Therefore for large dimensions such as 100 in our feature selection problem, this necessitates evaluating 100 neighbours at each iteration of the tabu search. In order to increase the efficiency of the tabu search algorithm, a random subset of neighbours was chosen. Although this introduced the additional control parameter *neighbourhood size*, this allowed the algorithm to run 3x faster than the original tabu search. A neighbourhood size of 30 was chosen to provide a balance between computation time and performance.

b) K flip neighbourhood

As an alternative to the one-flip neighbourhood, the k-flip neighbourhood is defined as all the bitstrings within a chosen k bit flips of the current solution. This introduces an additional hyperparameter k.

The k-flip neighbourhood was introduced as a further improvement to the tabu search algorithm due to the limitations on performance imposed by the selection of random neighbourhood subsets. While the random selection of neighbourhood subsets reduced the computational load and allowed the program to run within a reasonable amount of time, it resulted in a decreased amount of exploration and therefore a higher risk of converging to local optima.

In order to counteract this, a larger number of bit flips were applied when constructing the possible neighbourhood solutions. This allowed for increased exploration of the search space. Through experimentation, a k value of 3 allowed for the efficient use of random neighbourhood subsets while still ensuring good performance and high accuracies. Therefore the combined improvements of random subset selection and a k-flip neighbourhood with $k = 3$ resulted in significantly improved performance of the tabu search algorithm in terms of both efficiency and accuracy.

B.3. Objective function

The choice of the objective function forms a central part of the optimization algorithm as it guides the search. As the overall aim of the investigation is to reduce overfitting and thus improve generalisation, the objective function was chosen to be the accuracy of the ID3 classifier on the validation dataset. This was calculated by constructing the ID3 decision tree for each candidate subset using the training dataset and then evaluating the accuracy of the decision tree classifier on the unseen validation dataset. The accuracy metric was calculated as follows,

$$Accuracy = \frac{(TP + TN)}{(TN + FP + FN + TP)} \tag{4}$$

where TP, TN, FP and FN represent the number of true positives, true negatives, false positives and false negatives respectively. The validation accuracy provides an appropriate measure of generalization as it reports the classification performance on unseen data.

B.4. Recency-based tabu list

A recency-based tabu list was chosen over the frequency-based alternative due to the benefits offered in terms of memory use. The recency-based tabu list was implemented as a list of fixed size, where the most recently visited solution is appended to the list at each iteration and the first element of the list is removed if the maximum list size is exceeded. This effectively implements a sliding window of recently visited solutions. The size of the tabu list constituted a hyperparameter that requires tuning in order to prevent the search from revisiting suboptimal elements while still utilising a minimal amount of memory. Accordingly, a tabu list size of 8 was chosen through experimentation.

B.5 Control parameters

According to the chosen tabu search implementation, four control parameters required tuning. A critical parameter influencing performance constitutes the number of iterations. Since the number of fitness evaluations can be viewed as a computational limit, a maximum number of iterations of 10 was chosen to limit the computational time to approximately 25 mins to allow for a fair comparison with the genetic algorithm approach. For 10 iterations and a neighbourhood size of 30, approximately 300 function evaluations are required for one run of the tabu search algorithm. The final values chosen are summarized in Table I below.

TABLE I: Tabu Search Control Parameters

Technique	**Control Parameter**	**Final Value**
Tabu Search	Maximum iterations	10
	Tabu list size	8
	Neighbourhood size	30
	Bit flips k	3

C. Genetic Algorithm

In comparison to the tabu search algorithm, the design of a genetic algorithm contains significantly more variants and control parameters. The core components of a genetic algorithm constitute the encoding of the candidate solutions as chromosomes, choice of fitness function, selection of parents, reproduction to produce new solutions through crossover, the introduction of new genetic material by mutation and finally a replacement strategy to select the next generation. Therefore at each stage in the algorithm, several design choices need

to be made in accordance with the problem space and the appropriate amount of selective pressure.

C.1. Chromosome Representation

Genetic algorithms require the candidate solutions to a problem to be encoded as a chromosomal representation. Similarly to the tabu search algorithm, the most natural representation of the problem space in terms of a chromosome takes the form of a bitstring. Therefore we choose a binary representation in which each bit represents whether the feature is included in the subset or not. Consequently, our candidate solutions are encoded in the same way as in the case of the tabu search algorithm with the exception that for the tabu search, only one solution is chosen at each iteration. In contrast, for the genetic algorithm, multiple candidate solutions are initialised to form the population which morphs over the duration of the search. Accordingly, each individual in the population is initialized through random sampling of 1 or 0 for each of the 100 dimensions.

C.2. Parent Selection

The choice of selection operator plays a notable role in the amount of selective pressure applied to the population. Selective pressure refers to the speed at which the population will converge to the same solution. Too much selective pressure can have adverse consequences by decreasing the population diversity too quickly and therefore converging to sub-optimal solutions. However, inadequate selective pressure results in poor performance due to a lack of exploitation.

Two forms of popular selection operators were empirically compared in order to determine which was more appropriate for the constraints of the feature selection problem, namely roulette wheel selection and tournament selection.

a) Roulette wheel selection

Roulette wheel selection is a form of proportional selection as it calculates a probability distribution proportional to the fitness of the individuals which is then sampled to select individual chromosomes. This results in fitter individuals having a higher probability of being selected. The probability that individual $x_i(t)$ will be chosen is calculated as follows,

$$p_s(x_i(t)) = \frac{f_\gamma(x_i(t))}{\sum_{l=1}^{n_s} f_\gamma(x_l(t))} \tag{5}$$

where n_s is the number of individuals in the population. Roulette wheel selection is applied as summarised below, where each individual is assigned a segment along the span from 0 to 1 which corresponds to the proportional probability of the individual. A random number then determines which probability interval is chosen and therefore which individual is selected as a parent chromosome.

Algorithm 5 Roulette Wheel Selection

```
Let i = 0 point to the first chromosome
Calculate probability p(x_i)
Let sum = p(x_i)
Choose r ~ U(0, 1)
while sum < r do
  i = i + 1
  sum = sum + p(x_i)
end while
return x_i as the selected parent
```

In general, the main caveat of the roulette wheel approach is the tendency of a very fit individual to dominant selection and therefore cause the population to converge very quickly. However, this phenomenon was not observed in this case as the fitness values of the individuals were similar in scale and did not differ by a large magnitude. Due to this similarity in fitness level, roulette wheel selection performed sub-optimally in selecting the better individuals as parents, necessitating the exploration of a different selection operator.

b) Tournament selection

Tournament selection selects the best individual from a randomly sampled group of fixed size n_t. The amount of selective pressure applied by tournament selection is greatly dependent on the tournament size n_t.

This direct control over the amount of selective pressure applied to the search provided a strong advantage in our context. While roulette wheel selection failed to provide sufficient selective pressure to obtain desirable solutions, tournament selection resulted in a vast improvement in the final accuracy obtained. This high selective pressure was necessary to minimize computational time and expense as a lesser number of generations was required to obtain desirable results. Therefore tournament selection was chosen in favour of roulette wheel selection as the parent selection operator as it resulted in a highly significant improvement in the final accuracies obtained by the individuals in the population, as shown in Table II.

TABLE II: Empirical Study of Selection Operators

Operator	Variant	Accuracy	No. of Samples
Parent Selection	Roulette wheel	0.822	10
	Tournament	0.867	10

C.3. Bitstring Crossover

Since our chromosomes are encoded using a binary representation, a corresponding crossover technique suitable for binary representation was required. Therefore bitstring crossover was selected as the most appropriate technique. Bitstring crossover combines the genetic material of two parent chromosomes by producing offspring constructed from segments of each parent chromosomes. The number of segments is

determined by the number of cut points, where the position of each cut point is randomly selected.

Both one point and two point crossover were implemented to determine their effects on the search process. It was determined that two point crossover provided an advantage as it increased the explorative aspect of the genetic algorithm by allowing more complex combinations of parent chromosomes to be generated. Therefore two point crossover was chosen in the final configuration.

C.4. Random Mutation

Random mutation was applied by flipping each bit in the sequence according to the mutation probability p_m. Random mutation was chosen as a complement to the binary encoded chromosomes and applies mutation evenly across all bits.

C.5. Replacement Strategy

Two replacement strategies were considered. Initially, a form of *parent-offspring competition* was utilised by admitting offspring to the next generation if their fitness exceeded that of at least one parent, with the worst parent being removed from the population. However, using this strategy, the rate of change of the population was insufficient to produce desirable results as very few children were accepted. Therefore the genetic algorithm required many generations in order to converge to good solutions, causing high inefficiency.

To improve upon this, the replacement strategy was modified to accept offspring with fitness better than the worst individual in the population, thus removing the corresponding worst individual. This increased both the rate at which improved offspring were accepted and the chances of a profitable chromosome being chosen as a parent. The resulting algorithm forms a steady state genetic algorithms (SSGA), as there exists an overlap between the previous and next generation.

C.6. Fitness Function

To provide a representative measure of generalisation performance, the fitness function was chosen to be the validation accuracy as in the case of the tabu search algorithm. In a similar manner, the ID3 decision tree was constructed for the given feature subset using the training dataset and then the generalisation performance was measured by calculating the accuracy on the validation dataset.

C.7. Control parameters

a) Population size n_s The population size has a substantial impact on the performance of the genetic algorithm. It was determined that a population size of 30 was too large as it required increased computation expense and caused a slow rate of convergence. In contrast, a population of 20 was determined to be too small as the limited genetic diversity resulted in suboptimal accuracies. Therefore the population size was chosen to be 25 to provide a profitable balance between computation time and exploration ability.

b) Mutation rate p_m Initially the mutation rate was chosen to be 0.5. Although this helped to introduce more genetic material into the population, it caused the offspring to differ too significantly from the parent chromosomes and thus resulted in a low rate of acceptance of offspring into the successive generation. This caused a low improvement rate and sub-optimal results. Decreasing the mutation rate to 0.1 benefited the genetic algorithm as it increased the amount of exploitation and resulted in offspring more closely related to the parent chromosomes.

c) Crossover rate p_c The initial crossover rate of 0.8 was determined to be too low as an insufficient number of offspring were produced and accepted for each generation. Therefore the rate of improvement of the population was inadequate given the computational constraints. Increasing the crossover rate to 0.95 resulted in an increased number of accepted offspring and rate of improvement within the given computational time limit.

d) Maximum generations In order to enforce a similar computational limit on both the tabu search and genetic algorithm in order to ensure a fair comparison, the number of generations was chosen in keeping with the approximate time limit of 25 minutes. As the population size increases we decrease generations to maintain a reasonable run time. Therefore for the chosen population size of 25, the genetic algorithm was run for a maximum of 7 generations.

e) Tournament size n_t The tournament size has a direct relationship with the amount of selection pressure applied to the population. A tournament size that is too large results in a form of elitism as the best individual in the population will dominate selection. Alternatively, a tournament size that is too small will approach random selection. Accordingly, the tournament size was chosen to be a fifth of the total population size. The final values chosen are summarized in Table III below.

TABLE III: Genetic Algorithm Control Parameters

Technique	**Control Parameter**	**Final Value**
Genetic Algorithm	Maximum generations	7
	Population size n_s	25
	Crossover rate p_c	0.95
	Mutation rate p_m	0.1
	Tournament size n_t	5

D. Evaluation

In order to provide a statistically sound experimental design, the tabu search and genetic algorithm approaches were compared using the Mann-Whitney test with 30

simulations. The Mann-Whitney U test is a non-parametric test to determine whether two distributions differ in a statistically significant manner. The testing procedure and use of the three respective datasets is summarized as follows:

- Each optimization technique was run using the final configurations discussed above.
- During the search process, for each candidate feature subset, the ID3 decision tree was reconstructed with the new feature subset using the training dataset.
- The fitness for each optimization technique was determined by the classification accuracy on the validation set.
- Once the best subset had been found at the end of the search, the final test accuracy was computed using the best subset and the test dataset.
- This process was repeated 30 times for each optimization technique to obtain a statistically sufficient number of samples.

In this way, the test set was kept in isolation from the process of model building, tuning and optimization. In order to provide a comparable evaluation, the control parameters for each algorithm were chosen such that each algorithm ran for approximately the same amount of time and hence utilised approximately the same amount of function evaluations during the search. A computational time limit of approximately ≈ 25 minutes was chosen to ensure that all 30 simulations could be feasibly conducted for each algorithm. All algorithms were implemented in Python 3 and all experiments were run in a Google Colab environment.

IV. Results

The distribution of the performance results obtained for the tabu search and genetic algorithm respectively are shown in Fig. 1. While the summary statistics in Table IV provide an overview of the performance of each algorithm, an appropriate test it required to determine whether the difference in algorithm performance is statistically significant. From Fig. 1, it can be observed that the performance results of each algorithm are not normally distributed since they are skewed to the left, thus the results require a non-parametric statistical test for comparison. In addition, the samples are considered independent since the starting positions of each search are randomly generated. Hence the Mann-Whitney test is required in order to appropriately compare the performance results from the two optimization approaches.

TABLE IV: Summary Statistics of Performance Data

Algorithm	Mean	Variance	Minimum	Maximum
Tabu Search	0.905	3.644×10^{-4}	0.858	0.927
Genetic Algorithm	0.870	8.075×10^{-4}	0.792	0.917

The Mann-Whitney U Test is a null hypothesis test, used to detect differences between two independent data sets. Since our performance data is continuous, and we make no assumption about the distribution of our data, the Mann-Whitney test is appropriate for our context. We consider the

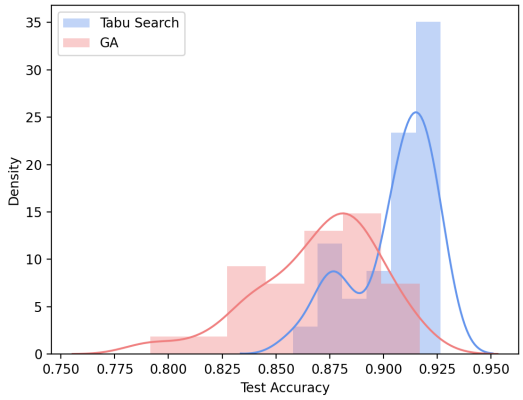


Fig. 1: Distribution of test accuracies obtained by the feature subsets selected using the genetic algorithm and tabu search respectively, with 30 simulations

null hypothesis $\mathbf{h_0}$ which states that the distributions of test accuracies obtained from the tabu search and genetic algorithm respectively are the same.

A. One-tailed Test

Based on our summary statistics which indicate that the tabu search algorithm produces superior results, we first consider a one-tailed test in which the alternative hypothesis $\mathbf{h_a}$ states that the the samples from the tabu search distribution tend to be larger than the samples from the genetic algorithm distribution. Therefore we seek to determine whether there is sufficient evidence to reject the null hypothesis. We calculate the U statistic to be 137.0, with a p-value of 1.91×10^{-6} as indicated in Table V. The critical value for a one-sided Mann-Whitney test at a significance level of $\alpha = 0.025$ and samples sizes $n_1 = n_2 = 30$ is given by 317.0. Since our U statistic is less than the critical value and our p-value is less than 0.025, we can conclude that the test accuracies obtained from the tabu search are greater than the test accuracies from our genetic algorithm at a significance level of 0.025.

B. Two-tailed Test

We next consider the more rigorous two-tailed test which makes no prior assumptions regarding direction. In this case, the alternative hypothesis $\mathbf{h_a}$ is defined such that the two distributions are statistically different. Therefore we seek to determine whether there is sufficient evidence to reject the null hypothesis and conclude that the distributions are not the same. Once again, we find the critical value for U such that if the observed value of U is less than or equal to the critical value, we reject $\mathbf{h_0}$ in favour of $\mathbf{h_a}$. The critical value can be obtained from the table according to the sample size $n_1 = n_2 = 30$ and our level of significance $\alpha = 0.05$ and is found to be 317.0. Our U statistic for the two-tailed test is calculated as 137.0 with a higher p-value of 3.81×10^{-6}. Therefore since $U_{statistic} < U_{critical}$ and $p < 0.05$, we can conclude that the two distributions are not the same.

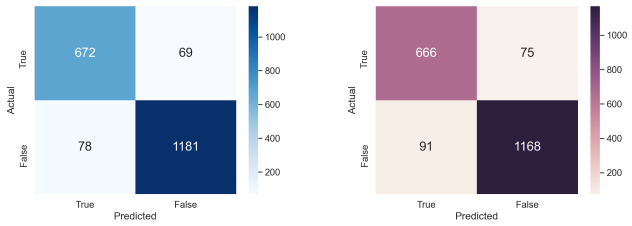
TABLE V: Mann-Whitney Test

Test	Simulation Parameter	Final Value
One-Sided Mann-Whitney	U Statistic	137.0
	p-value	1.91×10^{-6}
	Sample size n_1	30
Two-Sided Mann-Whitney	U Statistic	137.0
	p-value	3.81×10^{-6}
	Sample size n_2	30

For both the one-tailed and two-tailed tests, a very small p-value is observed. Since a lower p-value indicates stronger evidence against the null hypothesis, we have strong confidence in our result that the test accuracies obtained from the tabu search are greater than those of the genetic algorithm.

C. Classifier Performance

A confusion matrix was used to visualize the classification performance of the best subsets obtained from the tabu search and genetic algorithm respectively.



(a) Tabu Search (b) Genetic Algorithm

Fig. 2: Confusion matrix indicating classification performance for the best subsets selected by tabu search and genetic algorithm optimization.

Moreover, to give a fuller picture of classification performance rather than just the classification accuracy, the precision, sensitivity and specificity of the ID3 classifiers obtained using the best feature subsets were calculated as shown in Table VI below.

TABLE VI: Classification Performance Metrics

Algorithm	Accuracy	Precision	Sensitivity	Specificity
Tabu Search	0.927	0.896	0.907	0.938
Genetic Algorithm	0.917	0.880	0.899	0.928

It can be observed that the classification performance of the best subsets are relatively similar. However, the tabu search provides a more reliable and feasible solution to the optimization problem as the large majority of the feature subsets found by the tabu search attain test accuracies above that of the baseline. Without feature selection, the accuracy obtained on the test set is 88.1%. Therefore many of the solutions found by the genetic algorithm do not improve upon this baseline. Nevertheless, the solutions found by the genetic algorithm do allow a similar accuracy to be obtained with far fewer features, with most subsets being made up of between 50-55 features, therefore improving the efficiency of the algorithm.

A core factor influencing the results obtained is the computational time limit imposed on the search algorithms in order to perform a statistically significant test. The approach of this paper was to maximise the accuracy obtained in a relatively limited amount of function evaluations. Hence, it should be noted that potentially better accuracies could be obtained if the algorithms were allowed to run for longer. However, this would require re-tuning of the control parameters and may provide diminishing returns in some cases.

Another factor which could influence the outcome of the search algorithms is improved initialisation. Both search algorithms in the investigation were initialised randomly. This could be improved upon by initialising the algorithms in a more intelligent manner. Thus, further experimentation is recommended in terms of a longer computational time, further hyperparameter tuning and improved initialisation.

V. Conclusion

This investigation sought to determine whether tabu search or genetic algorithms provide a more appropriate solution to the optimization problem of feature selection, in the context of the ID3 decision tree classifier. Several key observations were made involving the specific design components of the search algorithms. Specifically, it was determined that the parent selection operator in the genetic algorithm plays a significant role in the final accuracies obtained, and that tournament selection provided an advantage over roulette wheel selection for our problem case.

In terms of overall search techniques, it was observed that the genetic algorithm could produce good results but required significantly more control parameter tuning and specific design choice than the tabu search algorithm. Moreover, it was determined that the accuracy results from the subsets obtained from the tabu search were statistically better than those of the genetic algorithm, as shown through both a one-tailed and two-tailed Mann-Whitney U test.

In addition, the tabu search provided more reliable results as the majority of feature subsets found by the tabu search improved upon the the baseline accuracy of 88.1% obtained by all 100 features. In contrast, many results from the genetic algorithm failed to improve upon this baseline. Notably, the average accuracy obtained by the feature subsets selected by tabu search was 90.5%, therefore attaining a 2.4% improvement over the baseline accuracy while simultaneously constructing a more efficient and faster decision tree. The performance of the genetic algorithm may be improved through better initialization and further hyperparameter tuning in order to fully harness the benefits of this approach, and is recommended for further study.

References

[1] Jinjie Huang, Yunze Cai, and Xiaoming Xu. A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern recognition letters*, 28(13):1825–1844, 2007.

[2] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.