

# A Hierarchical Probabilistic Divergent Search Applied to a Binary Classification

Senthil Murugan<sup>1</sup><sup>a</sup>, Enrique Naredo<sup>1</sup><sup>b</sup>, Douglas Mota Dias<sup>1,2</sup><sup>c</sup>, Conor Ryan<sup>1</sup><sup>d</sup>,  
Flaviano Godinez<sup>3</sup><sup>e</sup> and James Vincent Patten<sup>1</sup><sup>f</sup>

<sup>1</sup>University of Limerick, Limerick, Ireland

<sup>2</sup>Rio de Janeiro State University, Rio de Janeiro, Brazil

<sup>3</sup>Universidad Autónoma de Guerrero, Mexico

Keywords: Genetic Programming, Novelty Search, Classification.

Abstract: The trend in recent years of the scientific community on solving a wide range of problems through Artificial Intelligence has highlighted the benefits of open-ended search algorithms. In this paper we apply a probabilistic version for a divergent search algorithm in combination of a strategy to reduce the number of evaluations and computational effort by gathering the population from a Genetic Programming algorithm into groups and pruning the worst groups each certain number of generations. The combination proposed has shown encouraging results against a standard GP implementation on three binary classification problems, where the time taken to run an experiment is significantly reduced to only 5% of the total time from the standard approach while still maintaining, and indeed exceeding in the experimental results.

## 1 INTRODUCTION

Nowadays, the increasing interest of the scientific community in the Artificial Intelligence and addressing a wide range of real-world problems has led to increased interest in usage of open-ended search algorithms as a means of finding novel solutions. One example of an open-ended algorithm is Novelty Search (NS) (Lehman and Stanley, 2008). According to (Pattée and Sayama, 2019), there are many examples throughout the history of evolution on Earth showing that open-endedness is in fact a consequence of evolution. An open-ended approach has the goal to learn a repertoire of actions covering the whole space (Kim et al., 2021).


The original version of NS utilises the k-NN algorithm to explore the search space looking for novel behaviors and requires an archive to save previous behaviors in order to avoid backtracking. Defining the right parameter k and managing the archive could rep-


resent a problem to efficiently implementing NS. One approach to tackle these drawbacks is the Probabilistic NS (Naredo et al., 2016), which is the method we use in the work presented in this paper.


NS has been used successfully to address problems from different domains, such as the computational generation of video game content (Skjærseth et al., 2021), on one of the most impactful tasks in software development, namely bug repair (Trujillo et al., 2021), and a multi-objective brain storm optimization based on novelty search (MOBSO-NS) (XU et al., 2021) as well as many others.


More recently a new family of algorithms called quality-diversity algorithms (Pugh et al., 2016), which combine behavior space exploration with a global or local quality metric. An extensive study of these related algorithms is however beyond the scope of the research detailed in this paper.


The contributions of this paper are as follows: (i) the introduction of an implementation of the probabilistic version of novelty search (PNS) (Naredo et al., 2016) using DEAP (Fortin et al., 2012) a is a well known evolutionary computation framework, and more specifically details of an implementation that uses a Bernoulli distribution, (ii) the implementation of Pyramid Search (Ciesielski and Li, 2003) in


<sup>a</sup>  <https://orcid.org/0000-0003-0799-3079>

<sup>b</sup>  <https://orcid.org/0000-0001-9818-911X>

<sup>c</sup>  <https://orcid.org/0000-0002-1783-6352>

<sup>d</sup>  <https://orcid.org/0000-0002-7002-5815>

<sup>e</sup>  <https://orcid.org/0000-0001-5531-8989>

<sup>f</sup>  <https://orcid.org/0000-0002-1937-2774>

combination with OS and PNS, (iii) details of a case study conducted on the performance of the implementation on three binary classification problems, ranging from easy to hard.

The remainder of this paper is organized as follows: In Section 2 a review of the main background concepts. Next, Section 3 presents the experimental set-up, outlining all of the considered variants and performance measures. Section 4 presents and discusses the main experimental results of the described research problem. Finally, Section 5 presents the conclusions and future work derived from this research.

## 2 BACKGROUND

### 2.1 Genetic Programming

Genetic Programming (GP) (Koza, 1992) is an evolutionary algorithm which synthesizes computer programs. The evolution process initiates with a population of programs randomly created and assigning a fitness score according to the quality performance to solve the problem.

This score is then used to select individual programs following the Darwinian principle of natural selection to which genetic operators such as mutation and crossover are later applied, which drives the evolution of new programs in the hope that they are better than their parents.

Traditionally, evolutionary algorithms such as GP guide the search by rewarding individuals close to the predefined optimal fitness score target, which we refer to in this work as Objective-based search or for short OS. For instance, a predefined score target for a binary classification problem is to get 100% of classification rate.

In this work, we use binary classification problems and apply the Static Range Selection (SRS) described by Zhang and Smart (Zhang and Smart, 2006). A depiction of the SRS based classification can be found in figure 1. In this method, the goal is to evolve a function  $g : \mathfrak{R}^n \rightarrow \mathfrak{R}$  represented in the figure as a tree, which can map the vector of samples  $\mathbf{x}$  from the hyper-dimensional feature space into a 1-dimensional space. In the figure for visualizing purposes we use a 2D feature space  $(d_1, d_2)$ , meaning that each  $x_i = (d_1, d_2)$ , and the general case is  $x_i = (d_1, d_2, \dots, d_m)$ .

For a binary classification problem a typical threshold to use is zero to split in a natural way the one-dimensional space into two regions, where we have at the right the region  $\omega_1$  with positive numbers and at the left the region  $\omega_2$  with negative numbers. Predictions from the GP-classifiers are in these

regions, where  $\hat{\mathbf{y}}$  are the predictions to the true class  $\mathbf{y} \in \{0, 1\}$ . More specifically, for the binary case, we have as predictions a vector of 0s and 1s;  $\hat{\mathbf{y}} = [\hat{y}_1 = 0, \hat{y}_2 = 1, \dots, \hat{y}_m = 0]$ , for short  $\hat{\mathbf{y}} = [0, 1, \dots, 0]$ .

The performance of a GP-classifier is measured comparing the predictions  $\hat{\mathbf{y}}$  against the true class  $\mathbf{y}$  assigning 1 when the sample classification is correct and 0 otherwise, resulting in a vector  $\mathbf{b} = [b_1, b_2, \dots, b_m]$ , where the 1s are successes or hits, then the accuracy of a GP-classifier is computed as follows

$$Accuracy = \frac{\text{hits}}{\text{number of samples}} = \frac{\sum b = 1}{|\mathbf{b}|}, \quad (1)$$

where accuracy gives us an aggregated information of each GP-classifier through a single score value.

### 2.2 Novelty Search

Novelty search (NS) (Lehman and Stanley, 2008) is a diversity mechanism with a high search efficiency that helps the search to escape from local optima avoiding premature convergence.

NS is an alternative to OS, in which the objective function is replaced by a measurement of novelty and uniqueness (Lehman and Stanley, 2010). On one hand, we have OS rewarding individuals close to the target and on the other hand we have NS rewarding the most novel individuals. NS uses a behavior descriptor, which must capture the actions taken by the individual to make progress in the solution space. For a binary classification problem one feasible behavior descriptor is  $\mathbf{b}$  described in the previous section 2.1, whereas OS uses a score value, described in the Equation 1, where the higher score the better. Therefore, gradients in OS are towards objective, whereas the NS is towards behaviour.

The original measure of novelty (Lehman and Stanley, 2008) uses the k-nearest neighbors algorithm (k-NN), but instead of concern as to the compactness of the cluster where the individual is located, NS measures the sparseness  $S$  around each behavior  $\mathbf{b}_i$ . In other words, NS measures how different is the behavior from one individual against the behavior of its k-neighbors from  $\mathbf{b}_k$ , by computing the following equation:

$$S(\mathbf{b}_i) = \frac{1}{k} \sum_{l=1}^k dist(\mathbf{b}_i, \mathbf{b}_l), \quad (2)$$

where  $\mathbf{b}_l$  is the  $l$ th-nearest from  $k$ -neighbours respect to the average metric  $dist$ , which is a domain-dependent measure of the behavioural descriptor. A general interpretation of this measure is that a behavior  $\mathbf{b}$  is novel when located in a sparse region, but not

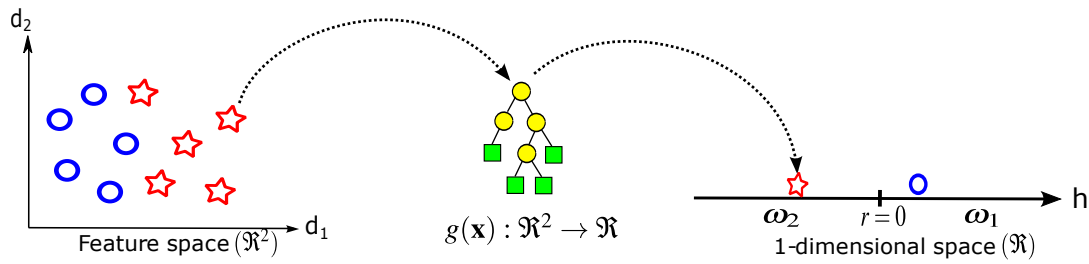


Figure 1: Graphical depiction of SRS-GPC.

novel when located in a dense region. This NS version is easy to implement, but requires the definition of  $k$  and this is computationally intensive for large training sets.

In order to avoid backtracking, the revisiting of regions that were previously explored in the search space, NS needs to save behaviors to an archive from the current generation. The behaviors from the current generation and from the archive can form a bigger set of behaviors to compute sparseness  $S$  as follows:

$$S(\mathbf{b}) = \frac{1}{k} \sum_{l=1}^k \text{dist}(\mathbf{b}_i, \mathbf{a}_l), \quad (3)$$

where  $\mathbf{a}_l$  is the  $l$ th-nearest from  $k$ -neighbours behaviors from both the actual generation  $t$  and the archive with behaviors from the previous generation  $t-1$ . For this version, when dealing with easy problems the amount of behaviors is usually manageable, but with more complicated problems a more considered management strategy is required. A well known method to manage the behaviors as a queue is the first-in first-out (FIFO). In this paper, we are not using the original version of NS.

### 2.3 Probabilistic Novelty Search

The original version of NS requires the tuning the parameter  $k$ , to decide what behaviors to store in the archive and to manage the archive. Probabilistic Novelty Search (PNS) (Naredo et al., 2016) aims to tackle these drawbacks, eliminating the parameter  $k$ , not using any archive, and as a consequence considerably reducing the computational effort without compromising the quality performance.

PNS is a probabilistic approach towards computing novelty using the same behavioral descriptor  $\mathbf{b}$  for NS. PNS assumes that a behaviour  $\mathbf{b}_i$  from an individual is independent of the behavior from any other individual, modeling the behaviors as random vectors. During the evolutionary search process, performed by GP, we have a population of  $B$  behaviors corresponding to the GP-classifiers,

$$\mathbf{B}^t = [\mathbf{b}]_j^t = [\mathbf{b}]_1^t, [\mathbf{b}]_2^t, \dots, [\mathbf{b}]_n^t, \quad (4)$$

where  $j$  is the index for each individual in the population, and  $t$  is the actual generation. Expanding the mathematical notation we have a matrix of behaviors each generation, as follows

$$\mathbf{B}^t = \begin{bmatrix} b_{1,1}^t & \dots & \dots & b_{1,n}^t \\ \vdots & \ddots & & \vdots \\ & & b_{i,j}^t & \\ \vdots & & \ddots & \vdots \\ b_{m,1}^t & \dots & \dots & b_{m,n}^t \end{bmatrix}. \quad (5)$$

Since we are using the hits, the matrix contains 0s and 1s. If we consider each outcome from the behavior descriptor as a random variable  $R$ , we can model the behaviors using a Bernoulli distribution, where the random variable  $R$  takes the success value 1 with probability  $p$ , and the failure value 0 with probability  $q = 1 - p$ . Then, the probability of success is given by  $P(R = 1) = p$ , and the probability of failure by  $P(R = 0) = q = 1 - p$ , where  $p$  and  $q$  depends on the number of samples and it is a well known fact that the estimator of  $p$  is the sample mean.

Taking the successful case where the random variable  $R = 1$ , we consider the hit from each behavior descriptor  $b_{i,j}^t = 1$ , in order to reduce the complexity of the mathematical notation and get a more readable notation, we simplify it by removing  $t$ , assuming that each behavior from the population is from the actual generation  $t$ .

As an illustrative example, say we had 7 samples from the dataset (rows) and 6 individuals (columns) in the population, then the order of the matrix  $\mathbf{B}^*$  is  $7 \times 6$ . As can be seen in the matrix  $\mathbf{B}$  shown in the Equation 6, in the first row it has all 0s, and in the last row has all 1s, meaning that the first is difficult to classify, whereas the last one is an easy sample to classify.

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \mathbf{p} = \begin{bmatrix} 0/6 \\ 1/6 \\ 3/6 \\ 4/6 \\ 2/6 \\ 4/6 \\ 6/6 \end{bmatrix}. \quad (6)$$

The probability  $p$  for each sample shown in  $\mathbf{B}$  to be correctly classified is given by the rate of number of successes  $R = 1$  over the number of observations and shown in the vector  $\mathbf{p}$ . For instance, the first sample has a probability of  $0/6 = 0$ , while the last one has  $6/6 = 1$  with 100% of successes.

We consider the product of the hit matrix,  $\mathbf{B}$  with its probability of hits,  $\mathbf{p}$ . To make the product possible, the vector is converted to a diagonal matrix  $\mathbf{D} = \text{diag}(\mathbf{p})$ , where each diagonal component is  $p_i$ . This way the product is  $\mathbf{P}^1 = \mathbf{DB}$ , where it only considers the hits cases, as follows:

$$\mathbf{P}^1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/6 & 0 & 0 \\ 0 & 3/6 & 3/6 & 0 & 0 & 3/6 \\ 4/6 & 0 & 4/6 & 4/6 & 4/6 & 0 \\ 0 & 2/6 & 2/6 & 0 & 0 & 0 \\ 4/6 & 0 & 0 & 4/6 & 4/6 & 4/6 \\ 6/6 & 6/6 & 6/6 & 6/6 & 6/6 & 6/6 \end{bmatrix}, \quad (7)$$

for the first individual we have  $[0, 0, 0, 4/6, 0, 4/6, 6/6]^T$  and summing up these probabilities we have  $14/6 = 2.33$ . Doing similar computation for the rest of the individuals results in a vector of accumulated expected values  $[2.33, 1.83, 2.50, 2.50, 2.33, 2.17]$  related to each individual.

The case where all the individuals successfully classify each sample  $6/6 = 1$  and having 7 samples in this example, then the maximum accumulated probability is 7, normalizing the previous vector we have  $[0.33, 0.26, 0.35, 0.35, 0.33, 0.31]$ . Finally, we have an aggregated probability information where PNS captures the probability differences among the individuals' behaviors.

$$PNS(\mathbf{b})_j^t = \sum_{i=1}^n \frac{1}{P_i(b_{i,j}^t) + \epsilon}, \quad (8)$$

where  $\epsilon$  is the total error of the probabilistic model against the true model. The novelty of each individual behavior is inversely proportional to the probability of predicting correctly (hit) each sample. Instead of managing an archive, we can save the parameters from the probability density function (PDF) for each sample and update them by using for example a convolution strategy between the previous and the actual PDFs.

## 2.4 Pyramid Search

There is a wide range of approaches in the literature to address evolutionary algorithms in a hierarchical way. For instance, a recent approach to address Deep Learning is proposed in (Houreh, et al., 2021), where the authors addresses the problem of evolving a neural network in a hierarchical way using GA as the search engine to automatically find the best architecture combination from a set of U-net architectures to solve the Retinal Blood Vessel Segmentation.

A novel hierarchical approach named Pyramid is proposed by (Ryan et al., 2020), where the main idea is to decompose a complex problem into subsets of simpler versions from the original and while scaling up to finally address the entire problem the population size is reduced keeping only the top individuals in each step.

Another hierarchical approach which shares part of the name with the previous approach is the Pyramid Search proposed by (Ciesielski and Li, 2003). In this approach the strategy is to split the population into groups and prune them a certain number of generations to reduce the number of evaluations and keeping competitive results. In this work, we use the latter approach applying it to solve a set of classification problems.

According to (Ciesielski and Li, 2003), Pyramid Search evolve optimal solutions using a reduced number of generations and individuals in the population by removing the worst performing groups. The Algorithm 1 gives the guidelines for the implementation of this method.

---

### Algorithm 1: Pyramid Search.

---

**Input:** The given problem and the solution representation.

**Output:** Best solution (with less groups of individuals).

```

1 Pop_size = No. of individuals in a group
2 Groups = No. of groups
3 Pruning_ratio = No. of least fit groups to be removed
4 Gens_to_prune = No. of generation to be pruned
5 Initialize_groups = Individuals(Pop_size)
6 while problem not solved or single group remains: do
7   Do evolution process
8   Prune.Groups(pruning_ratio * groups)
9 if problem not solved then
10  Continue iterations until single population exists
11 else
12  Problem solved or max generations is reached
13
```

---

In this approach the evolutionary process starts in the traditional manner by randomly creating an initial population, but in the next step, Pyramid Search splits the population into groups of fixed size. Each group is ranked based on its top individual, in other words,

the best individual from each group gives the representative fitness score to rank the group. The pruning process is based on the pruning ratio parameter, where the groups showing the lowest fitness value will be discarded, in other words, the groups bring their champions and the worst of them are executed with the whole group (all of them). The rest of the champions go to the next level with their entire group, this pruning process continues until the problem is solved or the maximum number of generations are reached.

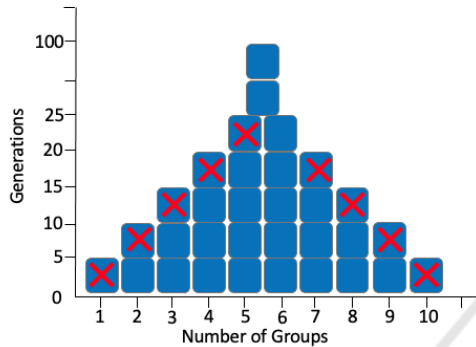


Figure 2: Graphical depiction of Pyramid Search.

In the figure 2, we can see a visual representation of the pruning process in the Pyramid Search. In the example every five generations a group competition takes place and the worst two groups are removed (represented by a red cross), an alternate way of looking at this is that the groups are sorted in such a way that the worst are placed at the ends. At the generation 20, only one group is removed and it remains until the evolutionary process stop.

### 3 EXPERIMENTAL SETUP

We designed a set of experiments based on binary classification problems to test both methods OS and PNS, and their combination with Pyramid, namely Py-OS and Py-PNS, particularly for the PNS methods we neither use an archive nor updated the PDF with previous distributions parameters.

#### 3.1 Datasets

Three datasets were selected: Cervical Cancer, Heart Disease, and Genetic Variation, sorted, from easy to hard.

**Dataset 1: Cervical Cancer.** The dataset (Repository, a) contains features such as age sex, smoke

Table 1: Binary Classification Datasets.

Datasets	Class	Vars	Samples
Cervical Cancer <sup>§</sup>	2	33	857
Heart Disease <sup>†</sup>	2	31	303
Genetic Variation <sup>§</sup>	2	57	3373

habits, STDs & other hormonal reading & the dependent variable represents the predictions of indicators or diagnosis of cervical cancer (Classes 0 or 1).

**Dataset 2: Heart Disease.** The dataset (Repository, b) has 13 independent variables which represents attributes such as age, sex, serum cholesterol, chest pain types, heart rate etc. The dependent variable refers to the presence of heart disease in the patient. Class 0 represents no/less chance of heart attack, and the Class 1 represents more chance of heart attack.

**Dataset 3: Genetic Variation Classification.** This is a public dataset containing annotations about human genetic variants (Clinvar, ). These variants are manually classified by clinical laboratories on a categorical spectrum ranging from benign, likely benign, uncertain significance, likely pathogenic, and pathogenic. Variants that have conflicting classifications can cause confusion when clinicians or researchers try to interpret whether the variant has an impact on the disease of a given patient. The objective is to predict whether a Clinical Var variant will have conflicting classifications. This is presented here as a binary classification problem, where each record in the dataset is a genetic variant.

#### 3.2 GP Parameters

Table 2 shows the parameters for GP in the top section and for the Pyramid Search at the bottom. The fitness function is the accuracy as shown in the Equation 1. Finally, all experiments were conducted using DEAP (Fortin et al., 2012).

### 4 EXPERIMENTAL RESULTS

The table 3 summarizes the experimental results, conducted over 30 experimental runs. The first column shows the datasets: Cervical Cancer, Heart Disease, and Genetic Variation, where the first is the easiest and the last the hardest. Second column shows the methods used: OS, PNS, Py-OS, Py-PNS, where OS stands for the Objective-based search and it is the traditional approach to rewards solutions closer to the target. The second method is PNS, which stands for



Table 2: GP and Pyramid Search parameters.

Parameter	Value
Runs	30
Population	200
Replacement	Tournament
Crossover	0.7
Function Set I	+, -, *, /, cos, sin
Function Set II	log, and, or ,not, if
<b>Pyramid Search</b>	
Pruning Ratio	0.2
Prune After Generations	5
No.of.Groups	10
Group Size	20

Probabilistic Novelty Search, which is a probabilistic approach to implement a divergent search. The rest are combinations of the first two methods with the Pyramid Search.

The experimental results start with the column 'Train', which stands for the average results in all runs using the training dataset, and 'Test' for the average results on test dataset. The column with 'Time' give us the information of the time spent by each run in average. Finally, the 'AvgSize' stands for the average size of the solutions considering the number of nodes in the GP-tree.

#### 4.1 Results in Training

Results shown in table 3, specifically for the Cervical Cancer dataset in training favors OS but does not have a statistical significant difference from PNS, even the standard deviation are similar among them. On the other hand, the combination using Pyramid Search shows results which are inferior to OS and PNS, but still in the same order, so both combinations show very good results. The results show PNS performs better in training for the Heart Disease with a significant difference against OS. The results in training for the Genetic Variation dataset, again place PNS as the champion, but followed closely by OS and the Pyramid combination are not that far.

figure 3 shows the convergence plot for the three datasets using OS and PNS, where it can be noted that the degree of difficulty for the methods to evolve good solutions for each of the datasets, the results clearly show that in training the easiest dataset is Cervical Cancer while the hardest is the Genetic Variation dataset. Figure 4 shows the convergence from the combination with Pyramid. Cervical Cancer dataset OS shows a better convergence can be seen and PNS struggles to continue evolving solutions. In the rest of the datasets while Py-OS demonstrates a tendency to evolve good solutions, they are not as good as those

evolved by OS and PNS, with Py-PSN getting trapped at a local optima in both datasets. A summarised performance in training is shown in the Figure 5, where Py-OS is worst than OS and PNS, and Py-PNS cannot find good solutions in training as the rest of the methods.

#### 4.2 Results in Test

Results in test shown for Cervical Cancer in table 3 interestingly favor Py-PNS, which shows the worst performance in training. Moreover, Py-PNS is the champion in test on all the datasets, this confirm the hypothesis that PNS can find novel solutions and in combination with Pyramid Search keeps best solutions with a reduced number of individuals from the original population.

The figure 6 shows the boxplots of the performance of each method in test, it can be observed in these that the median of Py-PSN is always above than the rest of the methods tested.

#### 4.3 Time Consumption

The experimental results shown in the table 3 regarding the time taken in average to run a single experiment from each method place Py-OS in the first place in all the datasets followed close by Py-PNS, interestingly PNS goes third in Cervical Cancer, but got the last position on the remaining datasets. This same trend is graphically observed in the Figure 7, where clearly both combinations using Pyramid in average take 5% of the time used by OS or PNS.

#### 4.4 Average Size

The experimental results shown in the table 3 regarding the average size from each method place Py-PNS in the first place followed by Py-OS, interestingly PNS ranks third in Cervical Cancer, but got the last place on the remaining datasets. The figure 8 shows the size grow for OS and PNS. For Cervical Cancer OS shows higher size in average than PNS, even though around the generation 85 both method show similar size average. For Heart Disease and Genetic Variation, OS shows a more flat grow rate, whereas PNS evolves more complex solutions on average. PNS does not get better performance than OS, but observing the boxplots from figure 6 can be observed that PNS gets a more diverse set of solutions than OS. The trend observed in the figure 9 regarding the average size of the solutions evolved by the methods in combination with Pyramid shows that Py-PNS evolve smaller solutions than Py-OS.

Table 3: Experimental results averaged over 30 runs, best results are bold.

Dataset	Method	Train	Test	Time	Avg Size
<i>Cervical Cancer</i>	OS	<b>0.9620</b> $\pm 0.003$	0.9580 $\pm 0.002$	98.92 $\pm 0.00$	5.00 $\pm 8.68$
	Py-OS	0.9609 $\pm 0.001$	0.9582 $\pm 0.002$	<b>5.82</b> $\pm 0.00$	4.71 $\pm 0.51$
	PNS	0.9619 $\pm 0.003$	0.9583 $\pm 0.003$	125.56 $\pm 0.00$	4.25 $\pm 3.86$
	Py-PNS	0.9575 $\pm 0.004$	<b>0.9621</b> $\pm 0.006$	6.52 $\pm 0.00$	<b>3.44</b> $\pm 0.37$
<i>Heart Disease</i>	OS	0.8221 $\pm 0.019$	0.7289 $\pm 0.059$	29.65 $\pm 0.00$	5.69 $\pm 3.44$
	Py-OS	0.7952 $\pm 0.020$	0.7194 $\pm 0.046$	<b>1.86</b> $\pm 0.00$	6.13 $\pm 1.93$
	PNS	<b>0.8272</b> $\pm 0.021$	0.7322 $\pm 0.053$	38.31 $\pm 0.00$	7.48 $\pm 5.31$
	Py-PNS	0.7588 $\pm 0.010$	<b>0.7362</b> $\pm 0.044$	2.25 $\pm 0.00$	<b>3.13</b> $\pm 0.29$
<i>Genetic Variation</i>	OS	0.7479 $\pm 0.001$	0.7222 $\pm 0.001$	411.30 $\pm 0.00$	9.17 $\pm 6.22$
	Py-OS	0.7472 $\pm 0.000$	0.7222 $\pm 0.002$	<b>25.36</b> $\pm 0.00$	6.80 $\pm 2.41$
	PNS	<b>0.7488</b> $\pm 0.001$	0.7224 $\pm 0.002$	533.92 $\pm 0.00$	13.01 $\pm 14.34$
	Py-PNS	0.7458 $\pm 0.000$	<b>0.7243</b> $\pm 0.000$	25.60 $\pm 0.00$	<b>2.82</b> $\pm 0.30$

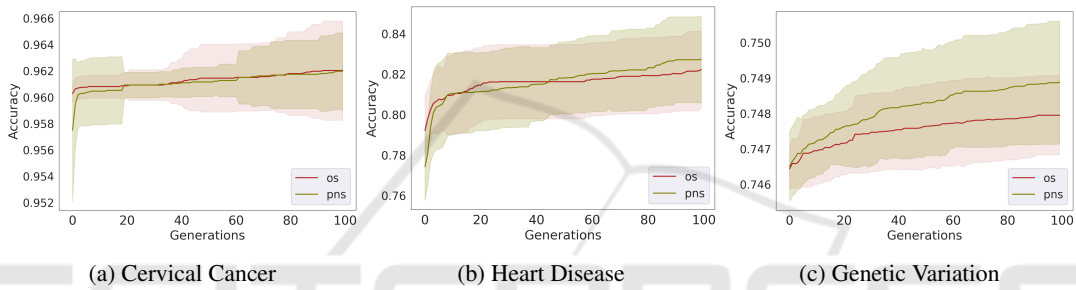


Figure 3: Convergence of training accuracy on datasets.

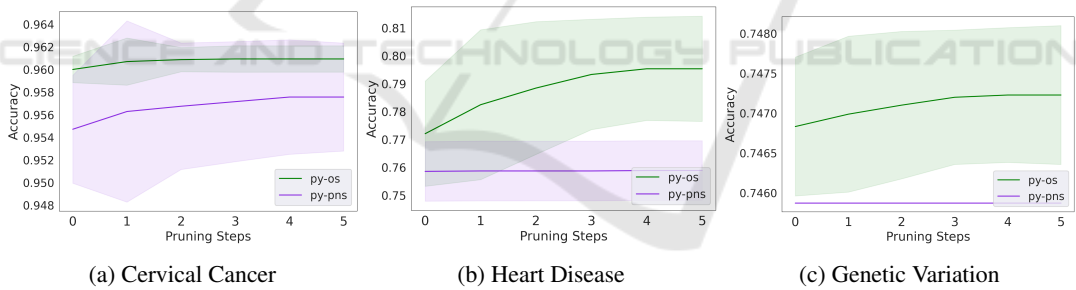


Figure 4: Pyramid Search - Convergence of training accuracy on datasets.

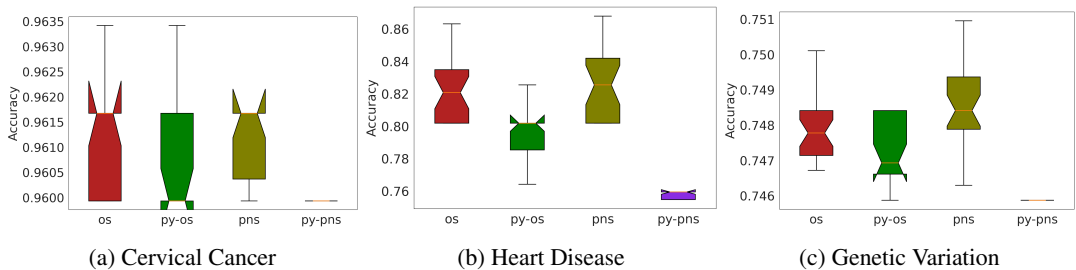


Figure 5: Accuracy on training data, box plot of all runs.

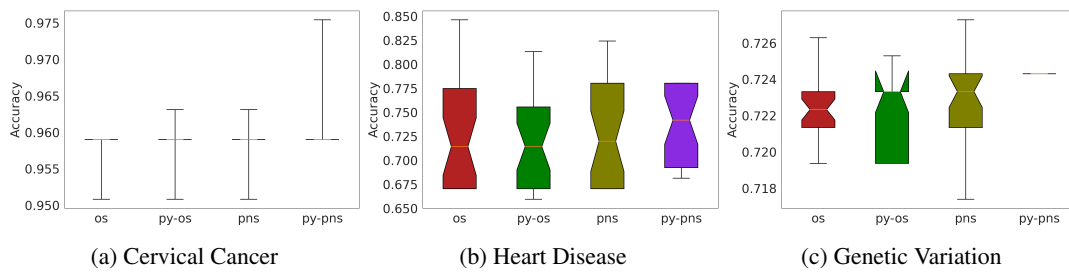


Figure 6: Accuracy on test data, box plot of all runs.

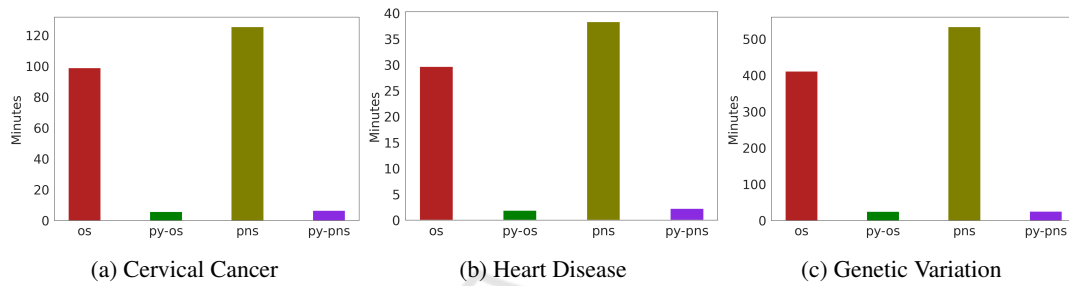


Figure 7: Time taken to run 30 runs on Datasets.

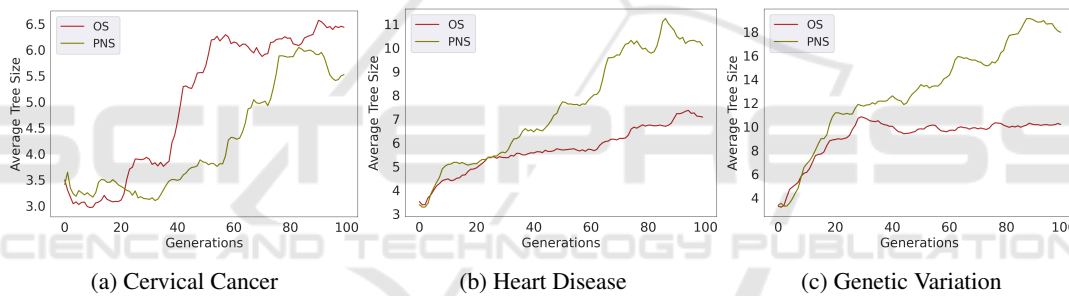


Figure 8: Average tree size of all 30 runs.

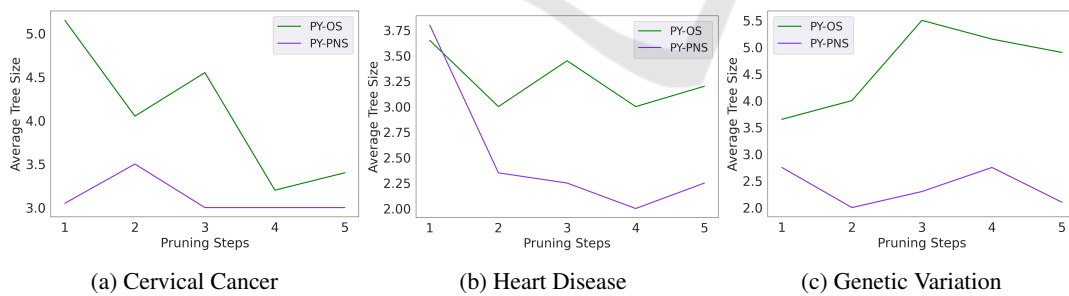


Figure 9: Pyramid Search - Average tree size of all 30 runs.

## 5 CONCLUSIONS

This paper introduced the implementation of the probabilistic version of novelty search, PNS for short, using a Bernoulli distribution, using the probabilities to successfully classify correctly the samples in the training set. Furthermore, the authors implemented

a combination of Pyramid Search with the traditional approach to guide the search named as Py-OS and the combination with PNS named as Py-PNS. As study case, three binary classification problems were used and the experimental results from 30 runs have shown that the combination with Pyramid gives good results and particularly Py-PNS is an interesting combination



to evolve diverse and optimal solutions reducing the average time for each experiment to 5% from the standard approach and getting even better results on some experiments.

In future work, the authors plan to compute novelty using PNS by (i) taking the probabilities for failure to classify the samples, (ii) considering both the success and failure. For future work we are considering Pyramid Search to automatically select the number of groups taking the population size and the number of generations as arguments.

## ACKNOWLEDGEMENTS

The authors are supported by Research Grants 13/RC/2094 and 16/IA/4605 from the Science Foundation Ireland and by Lero, the Irish Software Engineering Research Centre ([www.lero.ie](http://www.lero.ie)). The third is partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

## REFERENCES

- Ciesielski, V. and Li, X. (2003). Pyramid search: finding solutions for deceptive problems quickly in genetic programming. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 2, pages 936–943 Vol.2.
- Clinvar, N. Genetic variant classification dataset. [ftp://ftp.ncbi.nlm.nih.gov/pub/clinvar/vcf/\\_GRCh37/clinvar.vcf.gz](ftp://ftp.ncbi.nlm.nih.gov/pub/clinvar/vcf/_GRCh37/clinvar.vcf.gz). Online; accessed 28-September-2021.
- Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.
- Houreh, Y., Mahdinejad, M., Naredo, E., Dias, D., and Ryan, C. (2021). Hnas: Hyper neural architecture search for image segmentation. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART.*, pages 246–256. INSTICC, SciTePress.
- Kim, S., Coninx, A., and Doncieux, S. (2021). From exploration to control: Learning object manipulation skills through novelty search and local adaptation. *Robotics and Autonomous Systems*, 136:103710.
- Koza, J. R. (1992). *Genetic Programming - On the programming of Computers by Means of Natural Selection*. Complex adaptive systems. MIT Press.
- Lehman, J. and Stanley, K. (2008). Exploiting open-endedness to solve problems through the search for novelty. In Bullock, S., Noble, J., Watson, R., and Beaudou, M. A., editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 329–336. MIT Press, Cambridge, MA.
- Lehman, J. and Stanley, K. O. (2010). Efficiently evolving programs through the search for novelty. In Pelikan, M. and Branke, J., editors, *GECCO*, pages 837–844. ACM.
- Naredo, E., Trujillo, L., Legrand, P., Silva, S., and Muñoz, L. (2016). Evolving genetic programming classifiers with novelty search. *Information Sciences*, 369:347–367.
- Pattee, H. H. and Sayama, H. (2019). Evolved Open-Endedness, Not Open-Ended Evolution. *Artificial Life*, 25(1):4–8.
- Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). An extended study of quality diversity algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, GECCO '16 Companion*, page 19–20, New York, NY, USA. Association for Computing Machinery.
- Repository, U. M. L. Cervical cancer dataset. <https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29>. Online; accessed 28-September-2021.
- Repository, U. M. L. Heart disease dataset. <https://archive.ics.uci.edu/ml/datasets/heart+disease>. Online; accessed 28-September-2021.
- Ryan, C., Rafiq, A., and Naredo, E. (2020). Pyramid: A hierarchical approach to scaling down population size in genetic algorithms. pages 1–8.
- Skjærseth, E. H., Vinje, H., and Mengshoel, O. J. (2021). Novelty search for evolving interesting character mechanics for a two-player video game. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '21*, page 321–322, New York, NY, USA. Association for Computing Machinery.
- Trujillo, L., Villanueva, O. M., and Hernandez, D. E. (2021). A novel approach for search-based program repair. *IEEE Software*, 38(4):36–42.
- XU, Q., Pan, X., Wang, J., Wei, M., and Li, H. (2021). Multiobjective brain storm optimization community detection method based on novelty search. *Hindawi, Mathematical Problems in Engineering*, 2021:14. Article ID 5535881.
- Zhang, M. and Smart, W. (2006). Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recogn. Lett.*, 27(11):1266–1274.