NTNU
Kunnskap for en bedre verden

Department of Computer Science

IT3708 Bio-inspired Artificial Intelligence

Understanding Optimization Landscapes using Bio-inspired Algorithms

1 Introduction

The project focuses on feature selection within machine learning, which involves choosing a subset of variables from a dataset to enhance the performance of a predictive model while simultaneously reducing its complexity. Feature selection is critical for developing models that are more interpretable and, in some cases, computationally efficient. In this project, the goal is to analyze the structure of the feature selection problem through a bio-inspired approach. This not only addresses the problem itself but also provides a means to study the behavior of such algorithms as they search for the global optimum – in practical terms, the best combination of features across the fitness landscape.
The problem can be formally described by the following expression, which integrates two fundamental aspects:

- **Model Performance:** Measured by an error (for example, 1 – accuracy in the case of classification) obtained from a machine learning model trained on the selected feature subset.
- **Complexity Penalty**: A term that increases the cost of the solution as a function of the number of selected features, with the aim of favoring more parsimonious and interpretable solutions. Mathematically, the problem is expressed as:

$$h(x) = h_e(T(x)) + \varepsilon \cdot h_p(x)$$

  - $\mathbf{x}$ represents the selected feature subset.
  - $T(x)$ is the machine learning model trained on subset $\mathbf{x}$.
  - h_e denotes the error or loss of the model (e.g., classification error).
  - h_p represents the penalty associated with the number of features used.
  - ε is the weight that controls the relative importance of the penalty compared to the error.

  The goal is to **minimize** the function $h(x)$, thereby achieving an optimal compromise between model accuracy and solution simplicity.

2 Datasets

This project utilized three distinct datasets:

1. **ObesityDataSet** – This dataset consists of **16 features** and is used for a **multi-class classification** task, distinguishing between **7 different weight categories**.
2. **Winequality-white** – Containing **11 features**, this dataset originally posed a **regression problem**, where wines were rated on a scale from **1 to 10**. For this study, the task has been reformulated as a **multi-class classification problem**.
3. **MAGIC Gamma Telescope** – Comprising **10 features**, this dataset is derived from observations of **high-energy gamma particles** captured by an atmospheric Cherenkov telescope. It is employed for a **binary classification** task, distinguishing between gamma and hadron events.

Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transportation	Normal_Weight
Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportation	Normal_Weight
Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transportation	Normal_Weight
Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	Walking	Overweight_Level_I
Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transportation	Overweight_Level_II

Figure 1: Snippet of the first rows of the Obesity DataSet .

fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

Figure 2: Snippet of the first rows of the Winequality DataSet.

	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist	class
0	28.7967	16.0021	2.6449	0.3918	0.1982	27.7004	22.0110	-8.2027	40.0920	81.8828	g
1	31.6036	11.7235	2.5185	0.5303	0.3773	26.2722	23.8238	-9.9574	6.3609	205.2610	g
2	162.0520	136.0310	4.0612	0.0374	0.0187	116.7410	-64.8580	-45.2160	76.9600	256.7880	g
3	23.8172	9.5728	2.3385	0.6147	0.3922	27.2107	-6.4633	-7.1513	10.4490	116.7370	g
4	75.1362	30.9205	3.1611	0.3168	0.1832	-5.5277	28.5525	21.8393	4.6480	356.4620	g

Figure 3: Snippet of the first rows of the MAGIC Gamma Telescope DataSet.

3 Machine Learning Model

The selected machine learning model for this project is a Random Forest, configured with the following hyperparameters:

- `n_trees` = 100
- `max_depth` = 10
- `min_samples_split` = 10

The model's performance is evaluated using accuracy, which measures the proportion of correctly classified instances over the total number of instances.

To ensure reproducibility, the dataset was split into a training set (80%) and a test set (20%) using a fixed random seed (42).

4 Lookup Table Creation

To efficiently evaluate feature subsets across multiple population-based algorithms, I implemented a **precomputed lookup table**. This approach eliminates the need to retrain a model for every evaluation, significantly reducing computational overhead.

The process involved the following steps:

1. **Training a Machine Learning Model** – I selected a **Random Forest** as the model and trained it on all possible subsets of features from the dataset (2^n times).
2. **Performance Evaluation** – For each subset, I computed the **error metric** (accuracy-based loss).
3. **Storing Results** – Each subset's performance was recorded in a lookup table, mapping feature combinations to their respective error and penalty scores. The subsets were encoded as binary vectors, where each bit represents the inclusion (1) or exclusion (0) of a specific feature. For example, [0,0,0,0,0,0,0,0,0,0,0,1] represents the subset with only the 12th features (index 1), while [1,1,1,1,1,1,1,1,1,1,1,1] corresponds to the largest subset (index $2^n - 1 = 4095$ for $n = 12$).

5 Fitness Landscape Visualization

In this section, the results of the fitness landscape analysis are presented. The process consists of two main steps:

5.1 Local Minima Calculation

The number of local minima of the fitness function was calculated using a weight of 0.001, which controls the relative importance of the penalty term. This approach allowed us to identify the points where the algorithms could potentially become trapped.
The results of the local minima calculation for the fitness function, performed with $weight = 0.001$, are reported in Table 1.

Dataset	**Number of Local Minima**
ObesityDataSet	803
Winequality DataSet	26
MAGIC Gamma Telescope	15

Table 1: Number of local minima calculated for each dataset with $weight = 0.001$.

5.2 Fitness Landscape Visualization with t-SNE

To graphically represent the fitness landscape, the *t-Distributed Stochastic Neighbor Embedding* (t-SNE) technique was employed. This method is well-suited for the task as it reduces data dimensionality while preserving the local neighborhood structure, thereby facilitating cluster identification and a deeper understanding of the distribution of local minima.

The decision to use t-SNE is based on the following advantages:

- **Dimensionality Reduction:** t-SNE projects high-dimensional data into a two-dimensional space, making visualization more accessible.

- **Preservation of Local Structure:** It maintains proximity relationships between points, uncovering clusters and patterns in the fitness landscape.
- **Enhanced Interpretability:** The two-dimensional representation aids in comprehending the concentration and distribution of local minima.

To enhance the clarity of local optima visualization, fitness values were inverted, converting local minima into local maxima. The following images illustrate the t-SNE visualizations applied to the datasets under consideration.

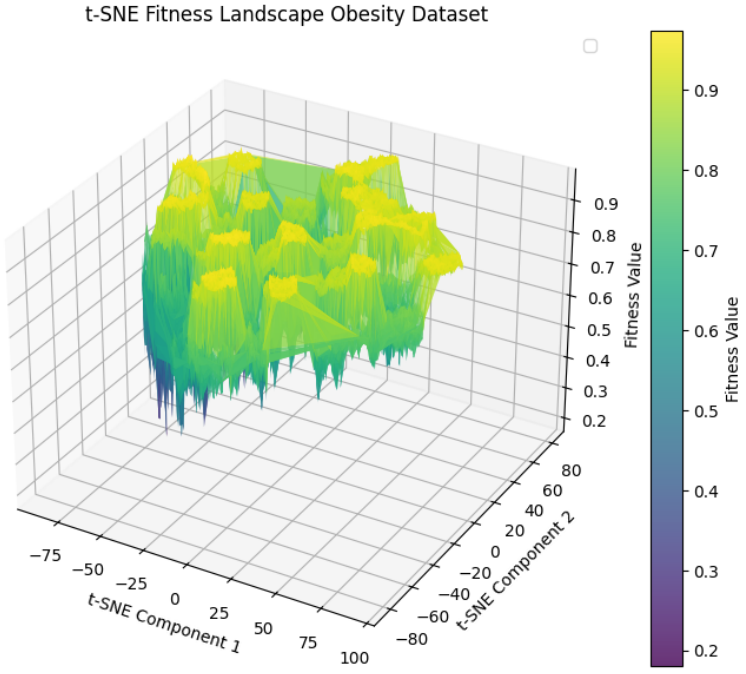

Figure 4: Visualization of the fitness landscape for the ObesityDataSet using t-SNE.

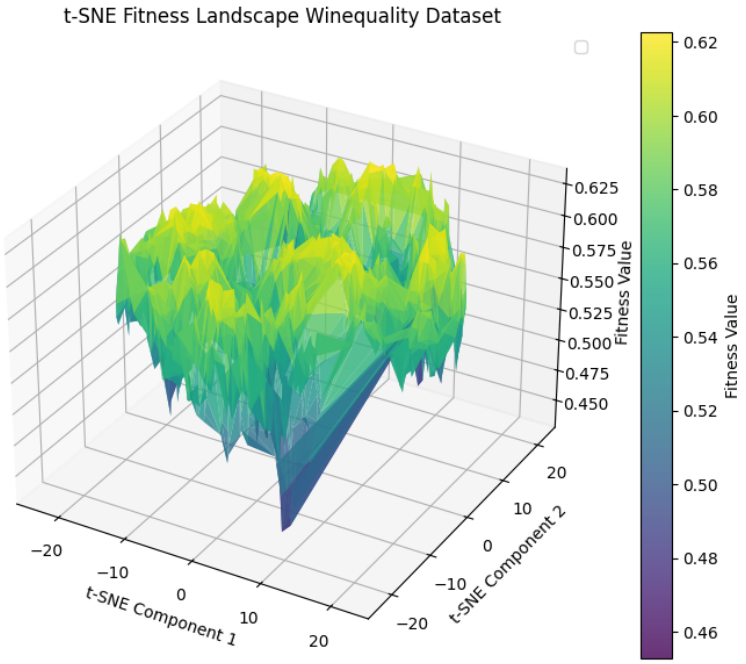

Figure 5: Visualization of the fitness landscape for the Winequality DataSet using t-SNE.

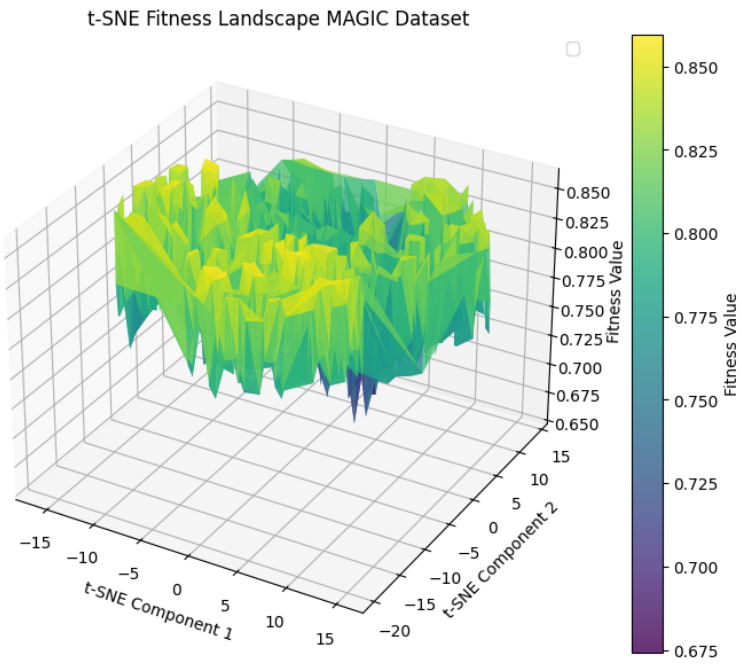

Figure 6: Visualization of the fitness landscape for the MAGIC Gamma Telescope using t-SNE.

6 Implementation of Bio-Inspired Algorithms

Three distinct bio-inspired algorithms were implemented, each representing a different class of optimization methods:

- **SGA (Simple Genetic Algorithm)** – A single-objective evolutionary algorithm.
- **NSGA-II (Non-dominated Sorting Genetic Algorithm II)** – A multi-objective evolutionary algorithm.
- **PSO (Particle Swarm Optimization)** – A swarm intelligence-based optimization algorithm.

All algorithms shared the same **random initialization** process and **terminated after 25 iterations**.

6.1 Simple Genetic Algorithm (SGA)

SGA is a classical evolutionary algorithm used for optimizing a single objective function. It operates through the principles of **selection, crossover, and mutation** to evolve a population over successive generations.

Implementation Details:

- **Selection:** The *tournament selection* method was used to choose parent solutions with *tournament size* $= 3$.
- **Crossover:** A *two-point crossover* operator was applied to produce offspring.
- **Mutation:** A *bit-flip mutation* was introduced to maintain diversity.
- **Survivor Selection:** The offsprings replace the previous generation.

6.2 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II is an advanced genetic algorithm designed for **multi-objective optimization**. It sorts solutions into **Pareto fronts**, preserving diversity through a **crowding distance mechanism**.

Implementation Details:

- **Parent Selection:** The *tournament selection* method was used to choose parent solutions with *tournament size* = 3.
- **Crossover & Mutation:** A *two-point crossover* and *bit-flip mutation* were applied.
- **Survivor Selection:** A *crowding distance mechanism* ensured diversity among solutions while minimizing two objectives:
  - The error produced by the random forest model based on the selected features.
  - The number of selected features.

6.3 Particle Swarm Optimization (PSO)

PSO is a **swarm intelligence algorithm** that mimics the collective behavior of birds and fish schools. The implementation steps included the following:

- **Velocity Update:** The velocity of each particle is updated based on the following factors:
  - Its own best-known position (*personal best*).
  - The best-known position in the entire swarm (*global best*).
  - The best-known position among its local neighbors, identified using the Hamming distance (*local best*).

  The velocity update follows the formula:

  $$v_{i,d} = wv_{i,d} + c_1 r_1 (p_{i,d} - x_{i,d}) + c_2 r_2 (g_d - x_{i,d}) + c_3 r_3 (l_d - x_{i,d})$$

  where:
  - w is the **inertia weight**, which determines the influence of the previous velocity.
  - c_1 is the **personal influence coefficient**, controlling the attraction to the best personal position.
  - c_2 is the **global influence coefficient**, controlling the attraction to the global best position.
  - c_3 is the **local influence coefficient**, which incorporates the best solution found among a particle's neighbors.
  - r_1, r_2, r_3 are random values sampled from a uniform distribution in $[0, 1]$.
  - $p_{i,d}$, g_d, and l_d represent the personal best, global best, and local best positions in dimension d, respectively.

  For the evaluation of the algorithm, the constants were set as follows: $w = 0.9$, $c_1 = 2.0$, $c_2 = 2.0$, $c_3 = 2.0$.
- **Position Update:** The new position is determined by applying the *sigmoid function* to the updated velocity:

  $$s(v_{i,d}) = \frac{1}{1 + e^{-v_{i,d}}}$$

  Each bit in the bitstring is updated by sampling a Bernoulli distribution with probability $s(v_{i,d})$.
- The local best neighbor is determined using a **Hamming distance threshold** of 2.

7 Experimentation and Results

Finally, we assess the performance of SGA, NSGA-II, and PSO on three datasets: ObesityDataSet, Winequality DataSet, and MAGIC Gamma Telescope. Each algorithm was executed 100 times with 30 iterations per run. The weight that controls the relative importance of the penalty is set to 0.001. Table 2 presents the key performance metrics.

Table 2: Performance Comparison of Optimization Algorithms over 100 runs, each with 30 iterations.

Algorithm	**ObesityDataSet**	**Winequality DataSet**	**MAGIC Gamma Telescope**
Best Fitness Target			
-	0.020	0.367	0.137
Mean Fitness ± Standard Deviation			
SGA	0.021 ± 0.001	0.367 ± 0.0007	0.137 ± 0.00
NSGA-II	0.021 ± 0.001	0.369 ± 0.0019	0.138 ± 0.00001
PSO	0.022 ± 0.0015	0.368 ± 0.0015	0.138 ± 0.0002
Success Rate (% of Runs Reaching Target Fitness)			
SGA	48%	95%	100%
NSGA-II	52%	50%	84%
PSO	43%	70%	82%
Convergence Speed (Iterations to Reach Best Fitness)			
SGA	17.6	10.2	6.7
NSGA-II	14.9	7	7.4
PSO	18.5	9.7	6.3

All three algorithms demonstrated good performance across the selected datasets, consistently finding solutions close to the optimal fitness value, even when the exact target was not reached. However, in several runs, especially in high-dimensional settings, the algorithms exhibited signs of premature convergence, getting stuck in local minima from which they could not escape.

As expected, when the dimensionality of the feature space decreases, the proportion of successful runs increases. Among the tested methods, NSGA-II appears to be the least effective, analyzing the Pareto fronts from ObesityDataSet, reveals that NSGA-II tends to rapidly push the population toward solutions with fewer features. While this might reduce complexity, it also limits exploration and increases the likelihood of getting trapped in suboptimal regions.

For instance, in the WineQualityDataSet, the globally optimal solution utilizes 8 features out of 11. Nevertheless, NSGA-II frequently converges to configurations with fewer than 5 features after just 15 iterations. This aggressive dimensionality reduction explains the faster convergence time but also highlights a key limitation in terms of exploration capabilities.

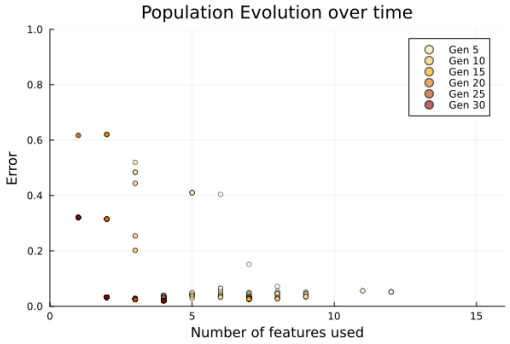

Figure 7: Pareto Front for WineQualityDataSet using NSGA-II.

In contrast, SGA and PSO achieved more balanced results, exhibiting greater robustness and a more consistent success rate across datasets. While PSO often converged slightly faster, SGA achieved a marginally higher number of successful runs overall. Both algorithms maintained a healthier exploration-exploitation trade-off, avoiding early convergence and delivering stable performance over repeated trials.