Understanding Optimization Landscapes using Bio-inspired Algorithms

Project 3 IT3708 – Spring 2025

Groups allowed: Groups are allowed and encouraged, up to 4 students. You can keep the same group from previous projects. All students in the group must be present during the demo day. All members of a group should sign up for the same time slot on demo day. For this project, you will do a conference-style presentation.

Guidance: Questions will be answered during lab hours or on Blackboard's discussion board.

Deadline: 28.04.2025

Introduction

The last two projects of this course have dealt with solving problems *practically*, that is, finding a specific solution that is close to the optimum. However, finding solutions is not all we do as professionals. Multiple times we need to create algorithms ourselves, or compare different solution methods. Sometimes we even need to find new problems to tackle. This project will let you practice these skills while looking at a familiar problem with a different, more theoretical mindset.

Lab Goals

- Create a surrogate model for an expensive fitness function, by combining machine learning and evolutionary computation.
- Optionally, create a visualization of the fitness landscape to highlight optima and get a general sense of the complexity of your fitness function
- Implement (or reuse) a **single objective evolutionary algorithm** of your choice
- Implement a **multi-objective evolutionary algorithm** of your choice
- Implement a **swarm intelligence optimization algorithm** of your choice
- Compare your algorithms and show your experimentation results
- Test your algorithms on an *unseen* landscape and try to find **multiple optima**

Problem Statement

You may have heard multiple times about the *No Free Lunch theorem in search and optimization*[1]: There is no such thing as a single best solver for all problems. Solvers do well

[1] https://en.wikipedia.org/wiki/No_free_lunch_in_search_and_optimization

because they happen to have built in assumptions which align with the structure of the problem. In this project we will try to capture the structure of the problem and study how different algorithms behave on the problem's terms.

The problem is the same as in Project 1: **feature selection**, which is a machine learning process where redundant and irrelevant features are removed from a dataset [1]. Evolutionary algorithms have been successfully used for the purpose of feature selection [10]. The selected features will often provide approximately as good, or better, results, and having data with fewer dimensions has some advantages when training a predictive model.

However, as you have learned throughout the course, there might be **multiple solutions** to this problem. What if we want to **find them all**? What if the solution that we found is **too complex to explain** to stakeholders? What if there is another combination of features with *good* results but with fewer columns involved?

You think for a moment and decide to study the **generalized feature selection problem**. You no longer want to find the best combination of columns for a particular dataset, you want to try to make arguments about datasets and the performance of models in general. What do different problems look like and what does it mean to navigate them? How do problems change depending on the nature of the datasets? Do some algorithms perform better on certain problems? Do some algorithms always trip up on certain problem structures? What characterizes problems which are easier and harder for different algorithms? After some deliberation, you arrive to a simple (but effective) model for reducing the complexity of the model (what we call *regularization*[2]):

Quality of a subset of features = The ML model performance (good) - How difficult it is to explain (bad)

Or in mathematical notation, using a canonical form with sums [8]:

$$h(x) = h_E(T(x)) + \epsilon h_P(x)\,, \qquad \text{(Eq. 1)}$$

where

- x is a subset of features (combination of columns)
- T is any given machine learning model trained on any given dataset using x
- h_E represents the error (in classification, or prediction, depending on T)
- h_P represents a penalty for the number of columns (or features) used, and
- ϵ represents the weight of the penalty.

Since $h(x)$ is about the error and a penalty, this is something you want to **minimize**.

Step 1—your choice: you decide which **dataset** to use, which **model** to use, which **error metric** to use, and **which penalty** to use. For example, a really simple one:

I want to use a decision tree classifier, on the iris dataset, using the classification error (1 - accuracy), and using the number of columns, multiplied by a small constant ϵ between 0 and 1.

[2] https://en.wikipedia.org/wiki/Regularization_(mathematics)

Project Tasks

Creating a Lookup Table

Since you want your experiments to be *general*, you decide that the best way to approach this is to make very comprehensive tests. However, comprehensive tests generally compare multiple population-based algorithms, and that means you will need to call this fitness function many, many times. Although you have done this in Project 1, training and then testing your feature subset at every evaluation will not work anymore. You need a faster alternative: precomputing a comprehensive, **lookup table**.

How do you do that? You need to train a machine learning model of your choice, on a dataset of your choice, **for all combinations of columns**, using the error metric and penalty of your choice. This exercise is very simple, but is computationally expensive. For all combinations of columns:

1. Train the machine learning model using the i-th subset of the columns.
2. Calculate the performance/error metric of your choice
3. Store the result in a row of a table

Assuming your dataset has *n* columns, that means that you will need to repeat this process 2^n times. So be mindful that *n* cannot be too large, as your table will end up having 2^n rows.
As you have probably gathered, calculating the penalty implies making a sum, so you can add the penalty $h_p(x)$ after you have stored your model performance $h_E(x)$.

Step 2—lookup table construction: using your choice of **dataset**, **ML model**, **error metric** and **penalty,** create three tables (one per dataset) that you can use as fitness functions for your algorithms. You are required to use either 3 datasets–1 model, or 1 dataset–3 models, for a total combined number of columns of $N \geq 27$. We highly suggest[3] you use values of $n < 20$. For example:

I want to use a decision tree on 3 datasets, with $n_1=7$, $n_2=10$ and $n_3=13$ columns respectively.

Where Do I Find Datasets?

There are several datasets repositories on the internet, for example UCI ML [2] and OpenML[4] [9]. If you are using Julia, there are packages which allow you to download datasets as tables directly, e.g., using *UCIData.jl* or *RDatasets.jl*.

Programming Language

You can use **any** programming language you want. There are plenty of Machine Learning libraries in Julia, Python, C++, Java, etc. However, for this lookup table step there is a lot to gain if you **parallelize your code** or **use a GPU** if you have access to one. This might impact your choice of programming language. If you are using Julia, take a look at *MLJ.jl* and *Distributed.jl*.

[3] Consider that 2^{20} = 1,048,576. You do not want to train a million neural networks for this project.
[4] https://www.openml.org/

Visualization (Optional Task)

Your tables represent three different problems, each with its unique search space, and each with its own **fitness landscape**. To start your analysis, you may want to visualize the landscape to help you pick suitable algorithms for your experimentation and understand the landscape better.

The Fitness Landscape[5] is an analogy to understand how complex it could be for any optimization algorithm to efficiently navigate the search space. Normally, one does not have access to the *entire picture* of a landscape, but luckily, you have precomputed absolutely all fitness values in Step 2—one for each combination of columns (or each point in the search space). Using the information provided by your lookup table, you can now make *a map*.

Step 3 (Optional)—Visualization: based on your **precomputed tables**, create a **visualization** for each landscape. There are several ways to do this, and we will cover some of them in our Landscape Analysis lecture. Choose any **visualization method** you like. If you decide to make a visualization, you are required to highlight each point considered as a **local optimum** in your landscape.

How you define a local optimum is also up to you. If you are using a bitstring, you might want to include only peaks, or you could consider plateaus as well. If you are using a continuous chromosome, you must define a threshold for optimality.

Programming Language

Again, you can use **any** programming language you want. There are plenty of professional plotting libraries in Julia, Python, C++, Java, etc. This step is quick and it is only done once, so you do not need to worry too much about computational speed. If you are using Julia, *Plots.jl* is a good first choice. If you want to have more control, take a look at *Makie.jl* and *CairoMakie.jl* for a more professional look.

Implementation

Now that your tables are ready, the difficult part has passed. From now on, it will be very easy to test as many evolutionary algorithms as you want, because you only need to look for the fitness value in the table. This is very fast, even if the table has a million elements.

Step 4—Implementation: Implement any three algorithms of your choice:

- One **single objective** evolutionary algorithm (for example., the one you used in Project 1)
- One **multi-objective** evolutionary algorithm
- One **swarm intelligence** optimization algorithm (for example, the one in EvoLP.jl)

Remember that your tables represent different fitness functions, so every time your algorithm needs to evaluate an individual, just look at the value in the corresponding table. One example of a selection of algorithms to implement could be:

I want to use the SGA, along with the NSGA-II, and PSO.

[5] Fitness landscape from 1:30 and onwards: FT018 - Function Landscapes and the Limits of AI for Bio

Experimentation and Presentation of Your Results

You are now ready to experiment and gather your results. With these results, you can draw conclusions about the algorithms you implemented and their search behavior, the datasets used, and even the machine learning models you trained. Did any of the landscapes prove to be more difficult for a specific algorithm?

You can annotate your results in a table, and create visuals to help convey your claims to your scientist colleagues. At least one of these elements must be presented during your demonstration.

Step 5—Experimentation and Comparison: Run your three algorithms on each landscape, and gather your results in a **table** or **plot**. Then, write your **conclusions** about them. What are the **hyper-parameters** you need to use on each algorithm in order to find the global optimum in your landscapes? Can the algorithms find **multiple optima**? Did you **control the parameters** during the run?

To make a fair comparison, you need to run the algorithms several times using similar hyperparameters. This is to eliminate any lucky initializations and see if the behavior observed is actually consistent. You are required to present statistical measures along with your results to ensure a fair comparison. For example:

I will run my algorithms 10 times each, store the results in a table, and calculate the average fitness value of the best performing individual across the runs. I will then create a table to present the results, along with the standard deviation of my sample of runs.

Testing your Algorithms on Unseen Instances

By playing around with your algorithms on different landscapes, you might now have an intuition on how they move through the search space. This was possible because doing Step 2 allowed you to have knowledge of the entire search space. You were navigating each landscape with the lights on. What would happen if the lights were off instead? And worse, what would happen if exploration were expensive?

A few days before the demo, we will provide you with three **feature selection problems** (or three landscapes—3 datasets $\times$ 1 model with specific hyperparameters) that will be computationally expensive to navigate. Your task will then be to find the optima. What are the subsets of features that minimize $h(x)$?

Step 6—Test Instances: Run any of your three algorithms on each of the test instances, and try to find the global optima. It is **very important** that you specify not only the optimal value $h(x^*)$, but also the point in the search space it corresponds to (the optimum, x^*).

Your grade will be based on *how close* you are from any of the global optima, which will be unknown to you. You will be searching blindly. However, the thresholds will be made public when you get the instances.

You are more than welcome to set and control the hyperparameters per instance. You may want to test different selection, mutation, crossover and survival operators, as well as trying out more advanced techniques to solve this challenge. Consider finding multiple optima and not only one.

Evaluation

As in project 2, you can get a **total of 35 points for this assignment (plus 3 extra points[6])**:

- Report and presentation of your work (10 points)
- Project tasks (25 points)

Project Tasks

The project tasks have the following weights:

- Step 1: Choose datasets and Machine Learning models – 0 points
- Step 2: Lookup Table Construction – 3 points
- Step 3 (Optional): Visualization – +3 points
- Step 4: Implementation – 9 points (3 points for each algorithm)
- Step 5: Experimentation and Comparison – 7 points
- Step 6: Test instances - up to 6 points, depending on the distance to the optimum

Note that Step 2: Lookup Table Construction can be very time consuming and memory intensive, depending on the datasets and models you choose. We highly suggest that you **start early**. Try parallelizing your code, and use the appropriate structures and types. You can also construct the tables in batches, concatenating everything when all batches are done. However, in case you and your team cannot complete **Step 2**, we have some precomputed tables available at Dataverse [7]. You can use some of these tables for steps 3-6, but you will not earn the points for Step 2.

Delivery Method and Deadline

You should hand in **2 deliverables in Blackboard** (a report, and your code), and give a **brief presentation** (10 minutes) during the **demo day**:

- **Code**: you should deliver a zip file with your code on Blackboard
- **Report**: you should deliver a group report (no more than 10 pages: 6 pages of the report and maximum 4 pages of appendix), summarizing the methodology, your experimental setup, your visualizations, the results of your experiments, your findings and your conclusions. Remember to include your references (including the datasets) and **cite them properly**.

The submission system will be closed on **April 28th 2025**. The **demo day** for Project 3 is also **April 28th 2025**. A signup schedule will be announced a week before. Please follow Blackboard for details. **Every student must submit a copy of their jointly developed code and report. You must attend the demo individually on the scheduled demo date. No early or late submission or demo will be entertained except in case of an emergency.**

[6] Valid only for this assignment.

Additional Reading on Landscape Analysis

We highly recommend taking a look at our paper on regularized feature selection, presented at Parallel Problem Solving from Nature ‘24 [8]. In the same conference, other researchers presented very similar results using different visualization methods [3].

If you want to know more about Landscape analysis in general, we highly recommend the Workshop from Gabriela Ochoa and Katherine Malan at GECCO ‘24 [5]. And, in case you want to delve deeper into visualization methods, you can read our PPSN ‘24 paper, or take a look at more expensive but highly descriptive methods like Local Optima Networks [4, 6]

References

[1] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *J Mach Learn Res* 3, null (March 2003), 1157–1182.

[2] Kelly Markelle, Rachel Longjohn, and Kolby Nottingham. UCI Machine Learning Repository. *UCI Machine Learning Repository*. Retrieved January 29, 2024 from https://archive.ics.uci.edu/

[3] Arnaud Liefooghe, Ryoji Tanabe, and Sébastien Verel. 2024. Contrasting the Landscapes of Feature Selection Under Different Machine Learning Models. In *Parallel Problem Solving from Nature – PPSN XVIII*, 2024. Springer Nature Switzerland, Cham, 360–376. https://doi.org/10.1007/978-3-031-70055-2_22

[4] Werner Mostert, Katherine M. Malan, Gabriela Ochoa, and Andries P. Engelbrecht. 2019. Insights into the Feature Selection Problem Using Local Optima Networks. In *Evolutionary Computation in Combinatorial Optimization* (*Lecture Notes in Computer Science*), 2019. Springer International Publishing, Cham, 147–162. https://doi.org/10.1007/978-3-030-16711-0_10

[5] Gabriela Ochoa and Katherine Malan. 2024. Landscape Analysis of Optimisation Problems and Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (*GECCO ’24 Companion*), August 01, 2024. Association for Computing Machinery, New York, NY, USA, 984–1004. https://doi.org/10.1145/3638530.3648421

[6] Gabriela Ochoa, Sébastien Verel, Fabio Daolio, and Marco Tomassini. 2014. Local Optima Networks: A New Model of Combinatorial Fitness Landscapes. In *Recent Advances in the Theory and Application of Fitness Landscapes*, Hendrik Richter and Andries Engelbrecht (eds.). Springer, Berlin, Heidelberg, 233–262. https://doi.org/10.1007/978-3-642-41888-4_9

[7] Xavier Fernando Cuauhtémoc Sánchez Diaz. 2024. Replication Data for: Regularized Feature Selection Landscapes: An Empirical Study of Multimodality. https://doi.org/10.18710/DQZKMX

[8] Xavier F. C. Sánchez-Díaz, Corentin Masson, and Ole Jakob Mengshoel. 2024. Regularized Feature Selection Landscapes: An Empirical Study of Multimodality. In *Parallel Problem Solving from Nature – PPSN XVIII*, September 07, 2024. Springer Nature Switzerland, Cham, 409–426. https://doi.org/10.1007/978-3-031-70055-2_25

[9] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2014. OpenML: networked science in machine learning. *ACM SIGKDD Explor. Newsl.* 15, 2 (June 2014), 49–60. https://doi.org/10.1145/2641190.2641198

[10] Bing Xue, Mengjie Zhang, Will N. Browne, and Xin Yao. 2016. A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Trans. Evol. Comput.* 20, 4 (August 2016), 606–626. https://doi.org/10.1109/TEVC.2015.2504420