UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Matheus Cândido Teixiera

Understanding the Fitness Landscape of AutoML Problems

Belo Horizonte
2023

Matheus Cândido Teixiera

Understanding the Fitness Landscape of AutoML Problems

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Gisele Lobo Pappa

Belo Horizonte
2023

Teixiera, Matheus Cândido

T266u Understanding the Fitness Landscape of AutoML problem [recurso eletrônico] / Matheus Cândido Teixiera — 2023.
1 recurso online (76 f. il, color.)

Orientadora: Gisele Lobo Pappa.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Departamento de Ciência da Computação, Instituto de Ciências Exatas
Referências: f. 67-72.

1. Computação – Teses. 2. Programação genética (Computação) – Teses. 3. Aprendizado de máquina– Teses. 4. Fitness Landscape – Teses. I. Pappa, Gisele Lobo. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. III.Título.

CDU 519.6*82 (043)

Ficha Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6/1510 Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Understanding the Fitness Landscape of AutoML Problems

MATHEUS CÂNDIDO TEIXEIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Profa. Gisele Lobo Pappa - Orientadora
Departamento de Ciência da Computação - UFMG

Prof Leonardo Vanneschi
NOVA Information Management School - Universidade Nova de Lisboa

Prof. Thiago Ferreira de Noronha
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 5 de dezembro de 2022.

Até aqui me ajudou o Senhor. (1 Samuel 7:12, adaptado)

Acknowledgments

First, I would like to thank God for the master's degree and for the countless miracles that allowed me to complete it. I also thank my supervisor, Gisele, for her help with several articles, my dissertation and in presenting one of our papers at a conference. I appreciate the support of my parents, siblings, beloved girlfriend, grandparents and uncles for helping and motivating me during this time. I thank all the support from LAIC, DCC, UFMG, and from my colleagues for the conquest.

"*The fitness landscape is one of the most influential concepts in evolutionary biology.*"
(Wright)

Resumo

A Fitness Landscape Analysis (FLA) engloba um conjunto de ferramentas utilizadas para compreender melhor as características do espaço de busca de um problema. Compreender o cenário de aptidão pode ajudar a melhorar os algoritmos existentes e fornecer informações sobre o problema. A maneira clássica de analisar as paisagens de aptidão é realizando uma análise exploratória da paisagem (ELA) usando um conjunto de métricas que alavancam características sobre o espaço, como correlação de distância de aptidão ou taxa de neutralidade. Outra forma de olhar para esses espaços é usando o Local Optima Network (LON), um grafo construído sobre a paisagem de fitness, onde cada nó representa um ótimo local no espaço de busca.

Este trabalho usa o FLA para entender melhor o cenário de adequação dos problemas do AutoML. Os principais desafios de compreensão dessas paisagens estão relacionados aos tipos de parâmetros envolvidos e à representação complexa das soluções (quando comparadas com a representação de soluções de outros problemas que tiveram suas paisagens estudadas). Embora a maioria das métricas seja desenvolvida primeiro para espaços contínuos ou combinatórios, os problemas de AutoML têm parâmetros categóricos, discretos e contínuos, muitos deles condicionais (ou seja, um hiperparâmetro está presente apenas se outro hiperparâmetro for previamente selecionado). A representação é de alguma forma hierárquica, já que alterações em algoritmos de classificação, por exemplo, tendem a ter um impacto muito maior no fitness do que alterar um único hiperparâmetro de algoritmo. Portanto, definir vizinhança neste espaço não é uma tarefa trivial. Com o objetivo de melhorar nosso entendimento dessas paisagens, definimos a paisagem de aptidão de um problema de AutoML e analisamos várias métricas FLA para verificar se elas são apropriadas para o espaço de busca induzido por problemas de AutoML. Primeiro, examinamos um conjunto de métricas tradicionais para ELA e realizamos experimentos em várias definições para verificar a robustez das métricas. Em seguida, analisamos o espaço sob a perspectiva da Rede de Ótimos Locais, que é particularmente boa para medir a neutralidade e rugosidade do espaço de busca.

Os resultados mostraram que, à primeira vista, os LONs são mais adequados para caracterizar esses espaços, que são multimodais e apresentam neutralidade acentuada, sendo um espaço desafiador para métodos locais.

Palavras-chave: Análise da Superfície da Fitness, Rede de Ótimos Locais, Aprendizagem de Máquina Automático

Abstract

Fitness Landscape Analysis (FLA) encompasses a set of tools used to better comprehend the characteristics of the search space of a problem. Understanding the fitness landscape can help improving existing algorithms and to give insights about the problem. The classical way of analyzing fitness landscapes is by performing an exploratory landscape analysis (ELA) using a set of metrics that leverage characteristics about the space, such as Fitness Distance Correlation or neutrality rate. Another way of looking at these spaces is by using Local Optima Network (LON), a graph built over the fitness landscape, where each node represents a local optima in the search space.

This work uses FLA to better understand the fitness landscape of AutoML problems. The main challenges of understanding these landscapes are related to the types of the parameters involved and the complex representation of the solutions (when compared to the solution representation of other problems that had their landscapes studied). While most metrics are first developed for either continuous or combinatorial spaces, AutoML problems have both categorical, discrete and continuous parameters, many of them conditional (i.e., one hyperparameter is only present if another hyperparameter is previously selected). The representation is somehow hierarchical, as changes in classification algorithms, for example, tend to have a much higher impact in fitness than changing a single algorithm hyperparameter. Hence, defining neighborhood in this space is not a trivial task. Aiming to improve our understand of these landscapes, we defined the fitness landscape of an AutoML problem and analyzed several FLA metrics to verify if they are appropriate for the search space induced by AutoML problems. First, we looked at set of traditional metrics for ELA and performed experiments on several definitions to verify the robustness of the metrics. Next, we analyzed the space from the perspective of Local Optima Network, which is particularly good to measure the neutrality and roughness of the search space.

The results showed that at first glance LONs are more appropriate to characterize these space, which are multimodal and present accentuated neutrality, being a challenging space for local methods.

Keywords: Fitness Landscape Analysis, Local Optima Network, Automated Machine Learning, Optimization

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In the past decade, the area of Automated Machine Learning (AutoML) has seen a boom of new algorithms and strategies to generate complete machine learning pipelines customized to the problem at hand [12]. A machine learning pipeline is defined by a set of operations for data preprocessing, data classification, and post-processing, which can be executed sequentially or in parallel. Without loss of generality, the classification task can be replaced by any other within the scope of machine learning, such as data regression or clustering. The methods are conceived to improve a measure of accuracy under constraints, as evaluating these pipelines demands a lot of computational power. Currently, all the big players have added to their cloud platforms methods for AutoML, including Google, Microsoft, and Amazon [6].

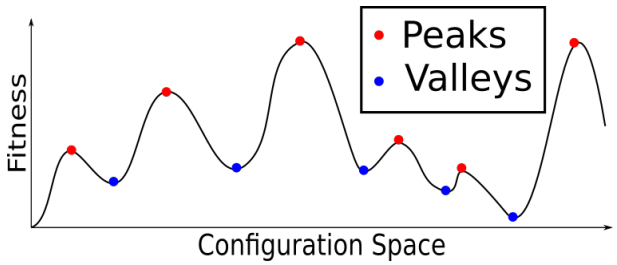
The methods used to generate these pipelines are mainly based on Bayesian optimization, evolutionary computation, or are hybrids [12]. In the past years the attention previously given to AutoML pipelines turned to searching for the best hyperparameters for artificial neural networks, known as Neural Architecture Search (NAS) [38]. However, one point that is not known in both areas is how is the structure of their search space, which can give hints on the best ways to explore it more effectively and efficiently.

As evolutionary computation are among the methods successfully used to explore these AutoML pipeline spaces, which started with simple methods such as TPOT [29] and Recipe [5] and evolved to more sophisticated approaches such as CMA-ES [9], it is natural to try to understand the fitness landscape these methods search on.

The fitness landscape of a problem is defined by a triplet $\{\mathcal{X}, \mathcal{N}, f\}$, where $\mathcal{X}$ is the set of valid solutions to the problem, $\mathcal{N}$ is the neighborhood of a solution $X \in \mathcal{X}$, and $f : \mathcal{X} \to \mathbb{R}$ is a fitness function [59]. By evaluating the solutions in $\mathcal{X}$ with f and having a notion of neighborhood, one can define the fitness landscape.

Knowing the fitness landscape, one can understand its characteristics, including roughness (the frequency of peaks and valleys), modality (number of global optima), and neutrality (adjacent regions in the configuration space where there is little or no fitness variation), which can be used as proxies to indicate the difficulty of the problem and can help developing search methods more appropriate to the landscape being explored. Figure 1.1 illustrates the fitness landscape of a one-dimensional configuration space (rep-

Figure 1.1: Fitness landscape of a 1D problem and its main characteristics.



Source: Elaborated by the author.

resented by the horizontal axis in the figure). In the figure, fitness is represented on the vertical axis and, in maximization problems, the higher the value, the better.

Fitness Landscape Analysis (FLA) [18] encompasses a set of tools used to better comprehend the characteristics of the search space. The classical way of analyzing fitness landscapes is by performing an exploratory landscape analysis (ELA) using a set of metrics that leverage characteristics about the space [23], such as Fitness Distance Correlation (FDC) – which measures the correlation between that fitness and distance to the global optimum [13]. Another way of looking at these spaces is by using Local Optima Network (LON) [28]. A LON is a graph built over the fitness landscape, where each node represents a local optima in the search space. Metrics coming from the field of complex network analysis [1] can then be extracted from this graph. LON has the advantage of generating a smaller space than the original, from which it can also extract metrics for measuring space roughness and neutrality, for example. The study of landscape can help to improve existing algorithms and to give insight about the problem [46].

The main challenges of understanding the fitness landscape of AutoML problems are related to the types of the parameters involved and the complex representation of the solutions (when compared to the solution representation of other problems that had their landscapes studied [28]). While most metrics are first developed for either continuous or combinatorial spaces [16], AutoML problems have both categorical, discrete and continuous parameters, many of them conditional (i.e., one hyperparameter is only present if another hyperparameter is previously selected). The representation is somehow hierarchical, as changes in classification algorithms, for example, tend to have a much higher impact in fitness than changing a single algorithm hyperparameter, e.g., changing the number of trees in a Random Forest has a lower impact than changing a Random Forest for a Naive Bayes. Hence, defining neighborhood is not a trivial task.

There are a few studies in the literature that have looked at the landscape of AutoML problems that generate machine learning pipelines. [8] performed an analysis of a subset of the search space explored by TPOT, and found many regions of very high fitness but prone to overfitting. [32] used fitness landscape analysis techniques to look at AutoML search spaces. They measured fitness distance correlation (FDC) [13] and

neutrality in a search space composed of machine learning pipelines for classification, and found FDC to be a poor metric for analyzing this kind of space. [34] analyzed the AutoML loss fitness landscape to verify if the scenario is generally unimodal or convex and if most of the hyperparameters are independent of each other. They also empirically demonstrate that FDC has limitations in characterizing certain spaces, which highlights the need for new methods with LON. In parallel, other studies have looked at the loss landscape of neural networks (NN) as architecture and hyperparameters are changed [35]. None of these studies are conclusive or perform more in-depth analysis of the fitness landscape being generated.

1.1 Objectives

Aiming to increase our understanding of the fitness landscapes of AutoML problems and how we can use their characteristics to make more informed choices of search methods, this thesis proposes different ways of representing fitness landscapes of AutoML problems and studies how these definitions affect the shape of the space and, consequently, the tools used to perform fitness landscape analysis. For that, we define a simplified search space of AutoML pipelines that is fully enumerable, where continuous attributes are discretized, and use it to shed light to two important research questions, defined next.

Research Question 1 (RQ1): *Are traditional metrics of fitness landscape analysis appropriate to characterize the search space of AutoML problems?*

The metrics used in the fitness landscape analysis are used to identify characteristics of space the space and are proxies for assessing problem difficulty. However, even the definition of the fitness landscape, in terms of representation and neighborhood, is a challenge. This first question, assuming we have a definition of fitness landscape, looks at whether traditional metrics of FLA used in the optimization literature, mostly focusing on classical problems, such as the Travel Salesman Problem (TSP) or Statisfability (SAT) problems are also adequate for AutoML. For that, we first define the AutoML problem as a generalization of the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem [7]. We then represent solutions (AutoML pipelines) as a syntax trees. This form of representation is natural giving the "hierarchical" nature of the problem, which arises from different dependencies between attributes. But it also brings challenges, such how to define of proximity/distance, as there is no "natural" way of measuring the distance between tree-like structures, specially considering they use different AutoML components that can be considered very similar or very different depending on

their semantic meaning.

A problem that arises from this decision is that many metrics depend on the concept of distance and the comparison of even identified configurations can generate completely different results. Therefore, the verification of the impact caused by the simple choice of distance is an important point in the analysis of the CASH problem and that highlights the complexity of this characteristic of a combinatorial problem with structured representation.

In order to answer this question, we empirically test different ways of defining solutions and measuring distances to verify how these changes impact the shape of the fitness landscape.

Research Question 2 (RQ2): *Compared to the traditional metrics of FLA, are Local Optima Networks (LONs) more suitable to characterize AutoML spaces?*

LON proposes a different interpretation of the search space, modeling it as a graph, and provide a more global view of the search space [28]. Unlike many traditional metrics, the representation of solutions as nodes of a graph makes this method suitable for dealing with combinatorial fitness landscapes, which is how we have simplified the AutoML space and made it enumerable.

In particular, we explore how LONs and its variations - namely Monotonic Local Optima Network (MLON) and Compressed Monotonic Local Optima Network (CMLON) [27] - can help explaining two characteristics that are almost a consensus in the AutoML that greatly affect problem difficulty [24, 32]: space neutrality - as sometimes solutions are within a narrow range of fitness values - and roughness.

1.2 Main Contributions

By answering the aforementioned questions, we list as our main contributions:

- A comparison of three methods for measuring distance between ML pipelines;
- An analysis of an enumerable AutoML search space using metrics from the FLA literature in 20 classification datasets, including both traditional metrics and those extracted from a LON and its variations
- A comparison of the analysis using both traditional FLA metrics and those extracted from a LON;

As a result of our work, we have so far published two papers related to the use of LONs for AutoML pipelines search space: [45] and [44]. The first focus on the analyses of roughness of the search space while the second deals with neutrality.

1.3 Text Organization

The reminder of this work is organized as follows. Chapter 2 formalizes and defines the problem of AutoML fitness landscapes. It also presents the main concept behind fitness landscape analysis (FLA), including exploratory landscape analysis and Local Optima Networks. Following, Chapter 3 details how the metrics if ELA were implemented in the context of AutoML, and analyzes different solution representations and different ways to calculate the distances between them. It shows the results for the metrics and discusses whether they are appropriate for AutoML. Next, Chapter 4 presents the concept of Local Optima Network (LON) as well as several metrics that can be extracted from this graph. It provides an analysis of these metrics in the context of AutoML problems. Finally, Chapter 5 presents the conclusions obtained from the experiments performed in the Chapters 3 and 4 and lists directions of future work.

Chapter 2

Background and related work

The concept of a fitness landscape was first introduced for characterizing the distribution of fitness values over the space of genotypes in (natural) evolution, and was later mapped onto a generic framework for optimization problems [16]. In optimization problems, the fitness landscape is defined by a tuple $(\mathcal{S}, f, N)$, where $\mathcal{S}$ is the set of all possible solutions (*i.e.* the search space), $f : \mathcal{S} \to \mathbb{R}$ is a function that attributes a real valued performance estimation for each solution in $\mathcal{S}$, and $N(x)$ is a notion of neighborhood between solutions, usually defined as a distance metric $N(x) = \{y \in \mathcal{S} \,|\, d(x, y) \leq \epsilon\}$ for a sufficiently small ϵ.

Knowing the fitness landscape of a problem, one can understand its characteristics, including roughness (the frequency of peaks and valleys), modality (the number of peaks), and neutrality (adjacent regions in the configuration space where there is little or no fitness variation). These characteristics may help defining which type of algorithm is the most advantageous for different types of problems. For example, search space with very neutral regions, i.e., regions of the search space where all solutions share the same value of fitness, should not be explored by gradient-based algorithms, as they are likely to be stuck in these plateaus.

With this objective, fitness landscape analysis (FLA) encompasses a set of metrics and methods to extract these features from the fitness landscape, and it has shown to be very useful for characterizing and analyzing search spaces. The classical way of understanding fitness landscapes is by using a set of metrics that leverage characteristics about the space, as discussed in this chapter. Another way of looking at these spaces is by using Local Optima Network (LON) [28].

This chapter first defines the problem of fitness landscape analysis of AutoML problems from an optimization point of view. It then reviews the two main approaches for fitness landscape analysis aforementioned. We end the chapter with a review of related work.

2.1 Problem Definition

The problem of AutoML can be formally defined as a generalization of the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem [7]. In its original definition, given a set $\mathcal{A} = \{A^{(1)}, A^{(2)}, \ldots, A^{(k)}\}$ of learning algorithms, where each algorithm $A^{(j)}$ has a hyperparameter space $\Lambda^{(j)} = \{\lambda^{(1)}, ..., \lambda^{(S)}\}$, defined from the full set of algorithm's hyper-parameters $\mathbf{\Omega}$, the CASH problem is defined as in Eq. 2.1[1].

$$A^*_{\lambda^*} = \underset{A^{(j)} \in \mathcal{A}, \lambda \in \Lambda^{(i)}}{\operatorname{argmax}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{F}\left(\mathcal{A}^{(j)}_{\lambda}, \ \mathcal{D}^{(i)}_{train}, \mathcal{D}^{(i)}_{valid}\right) \tag{2.1}$$

where $\mathcal{F}(A^{(j)}_{\lambda}, \mathcal{D}^{(i)}_{train}, \mathcal{D}^{(i)}_{valid})$ is the gain achieved when a learning algorithm A, with hyperparameters Λ, is trained and validated on disjoint training and validation sets $\mathcal{D}^{(i)}_{train}$ and $\mathcal{D}^{(i)}_{valid}$, respectively, on each partition $1 \leq i \leq k$ of a k-fold cross-validation procedure.

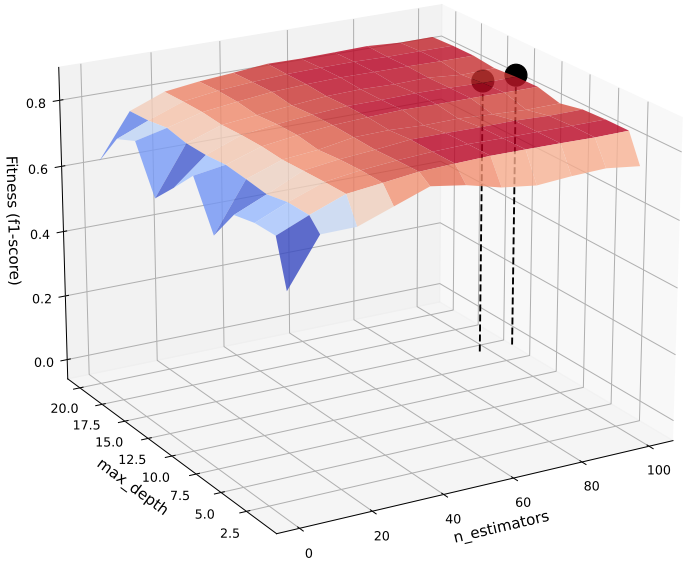
A generalization can be made if we replace $\mathcal{A}$ by a set of pipelines $\mathcal{P} = \{P^{(1)}, ..., P^{(V)}\}$, which includes a subset of algorithms from $\mathcal{A}$ and their respective set of hyperparameters $\Gamma^{(i)} = \{\Lambda^{(1)}, ..., \Lambda^{(S)}\}$, represented by the full set $\mathbf{\Psi}$, as defined in Eq. 2.2.

$$\mathbf{P}^*_{\mathbf{\Gamma}^*} = \underset{\mathcal{P}^{(i)} \subseteq \mathbf{P}, \mathbf{\Gamma^{(i)}} \subseteq \mathbf{\Psi}}{\operatorname{argmax}} \ \frac{1}{K} \cdot \sum_{j=1}^{K} \mathcal{F}(\mathbf{P^{(i)}}_{\mathbf{\Gamma^{(i)}}}, D^{(j)}_{train}, D^{(j)}_{valid}) \tag{2.2}$$

We need to define the fitness landscape of our problem according to this definition. Considering the tuple $(\mathcal{S}, f, N)$, where $\mathcal{S}$ is the search space and is defined by the set of the pipelines $\mathcal{P}$, the fitness f is a measure of algorithms performance in $D^{(j)}_{valid}$, such as accuracy, and the most difficult thing is to define the concept of neighbor solutions N, as discussed later in this chapter. As an example, Figure 2.1 shows the fitness landscape of a very simple problem, where the objective is to optimize the f1-score (fitness) of the search space defined by the set of algorithm $\mathcal{A} = \{\text{Random Forest}\}$ and hyperparameters $\Lambda = \{$`n_estimators`, `max_depth`$\}$. As we are optimizing only two parameters, it is possible to visualize the surface in a figure, which is not possible in higher dimensional spaces. The highlighted points are the global optima, i.e., the solutions that obtained the highest fitness in this space and occur when `n_estimators` is equal to 90 or 100 and `max_depth` is equal to 12. The challenge of this task is to maximize the fitness while minimizing the number of evaluations. Note that, in general, AutoML algorithms do not have the information of the global structure of the fitness landscape, as the number of combinations of the parameters available is too high.

[1] The original definition casts the problem as a minimization one. Here we replace the loss function by a gain function.

Figure 2.1: Example of fitness landscape considering two hyperparameters of the Random Forest algorithm.



Source: Elaborated by the author.

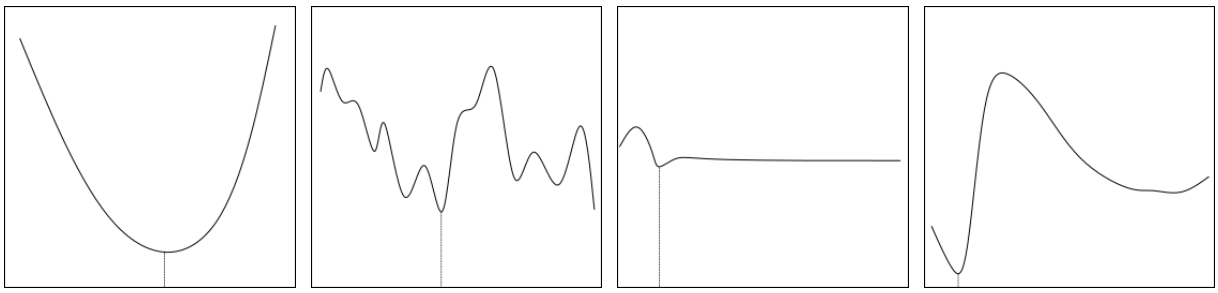
2.2 Standard Metrics for Fitness Landscape Analysis

Having a fitness landscape defined, the classical way to analyze it is by looking at a set of measures capable of giving us insights on its characteristics. In this context, we need to consider what types of characteristics we can analyze and what they mean regarding the structure of the space.

Note that analyzing fitness landscapes is difficult due to their size and number of dimensions. Because of that, researchers have looked at their main characteristics and how they vary. [18] indicates that roughness, deception and neutrality are characteristic of the difficulty of the fitness landscape of optimization problems. Figure 2.2 presents examples of each of these characteristics. Figure 2.2a shows a smooth landscape where an algorithm can use the gradient to direct the search towards the optimal solution. Figure 2.2b illustrates a rough landscape. Roughness indicates the presence of many local optima in space and is a feature studied in many works [14, 19, 42], as it can make exploration of the search space difficult because it provides little information on how to find some global optimum. Figure 2.2c gives an example of a neutral landscape (the flat region), where the derivative tends to zero. Neutrality indicates the presence of regions

where there is little variation in fitness between neighboring solutions [30, 55, 50, 53, 54]. This property can affect algorithms that use local space information to decide how to proceed. Finally, Figure 2.2d presents an example of a deceptive landscape. Problems with this type of fitness landscape provide false information leading search algorithms to deviate from the global optimum.

Figure 2.2: Examples of fitness landscapes with different characteristics.



(a) Smooth landscape (b) Rugged landscape (c) Neutral landscape (d) Deceptive landscape

Source: Elaborated by the author.

Many metrics have been previously defined in the literature [13, 15, 17, 37, 18]. The work of [60] addresses several application areas and a range of metrics being used to analyze the fitness landscape of optimization problems. They indicate that the method usually used by many of these metrics for constructing the configuration space – obtained through a random sampling of the real space – may prevent achieving a more accurate FLA result. In order to mitigate this problem, the authors indicate the use of more than one metric can help to understand the characteristics of the landscape more precisely. For example, metrics that measure neutrality do not provide information about the roughness of the fitness landscape, so using metrics to measure both characteristics may be more appropriate.

Here we focus on four of them: Fitness Distance Correlation (FDC) [13], Dispersion Metric (DM) [15], First Entropic Measure (FEM) [17] and Neutrality Rate [37]. Note that DM and FEM were originally designed for continuous landscapes, and suffered adaptions to deal with the generated combinatorial space analyzed in this work. These classical metrics were adopted because the FDC and Neutrality Rate have already been used in similar works in the literature and the FEM and DM were used for experimental purposes to analyze the roughness and concentration of solutions at specific points in space, respectively.

It is important to mention that the objective of that chapter is to analyze the space metrics popularly used in the literature on the set of datasets studied and to discuss the characteristics they indicate about the problem. However, during the development of this work, several other works in the literature appear criticizing the quality of some of these metrics present in this section [32, 34]. This will be further discussed in Section 2.4.

2.2.1 Fitness Distance Correlation (FDC)

Fitness Distance Correlation (FDC) [13] is a metric widely used in papers about FLA and was already used in the context of AutoML [24, 32, 46]. The idea behind this metric is that landscapes that have a positive correlation between distance and fitness from a global optimum are proportionately easy to optimize [39]. It was originally proposed by [13], but had the limitation of depending on the knowledge of the global optimum, which is often not available. To work around this limitation, FDC_e is provided as an alternative.

In FDC_e, a random sample of n solutions, $s_1, \dots s_n$, with their corresponding fitness, $F = \{\mathcal{F}(s_1), \dots \mathcal{F}(s_n)\}$, are generated. Then the solution with the highest fitness, $s*$, is identified and the distance is obtained using the global optimum of the sample $D = \{d(s_1, s*), \dots, d(s_n, s*)\}$. The final metric can be expressed as follows:

$$\text{FDC}_e = \frac{\text{cov}(F, D)}{\sigma(F) \cdot \sigma(D)}$$

where cov() represents the covariance and σ, the standard deviation.

FDC can be interpreted as follows: a positive and high FDC indicates that seeking solutions towards configurations where fitness increases can be a good optimization strategy; while in problems where the FDC is close to 0 or negative, this fitness-guiding strategy may not be advantageous.

FDC has been previously used to characterize the landscape of both combinatorial [32, 46, 20, 4] and continuous landscapes [36].

2.2.2 Dispersion Metric (DM)

The Dispersion Metric [15] measures the average distance between the top $p\%$ solutions from a set of randomly sampled points using a uniform distribution. The idea is to measure how close or dispersed the solutions with the highest fitness are, and it is calculated as follows: first, we sample a fixed-length set of solutions s_v from the search space and evaluate their fitness. We then rank the solutions by their fitness value, and the best $s_b = s_v \times p\%$ solutions are selected, and $disp$ is the average distance between these solutions.

This metric has been used to characterize continuous fitness landscapes [22, 23], but it was not necessary to change the algorithm to apply it to the combinatorial space, since the distance is calculated directly between two pairs of solutions.

2.2.3 First Entropic Measure (FEM)

The metric is calculated over a random walk $P = p_1 p_2 ... p_n$ in the configuration space. For each triple $p_{k-1}p_k p_{k+1} \in P$, a symbol $s_k \in \mathcal{S} = \{\bar{1}, 0, 1\}$ represents the difference between the fitness of two consecutive solutions in the random walk [17]. That is, a sequence of symbols are used to analyze the characteristics of the fitness landscape and they are defined through the difference between the fitness f_i of the i-th configuration and the fitness f_{i+1} of $i+1$-th configuration, that is, the symbol that represents the walk $p_{i-1}p_i p_{i+1}$ are defined by the function 2.3:

$$\mathcal{S}_i = \Psi_{f_t}(i, \epsilon) = \begin{cases} 1, & \text{if } f_{i+1} - f_i < \epsilon \\ 0, & \text{if } |f_i - f_{i+1}| \leq \epsilon \\ \bar{1}, & \text{if } f_i - f_{i+1} > \epsilon \end{cases} \tag{2.3}$$

where ϵ indicates the tolerance for considering a sequence to be neutral or not. As can be deduced from Equation 2.3, a very large value of ϵ can generate a completely neutral surface and a very small value can represent an extremely rough surface. Therefore, to choose an appropriate value for ϵ, [17] proposes the constant ϵ^* equals the largest difference present in the path. The pattern of symbols indicates roughness and neutrality characteristics of the fitness landscape.

[17] defines a method for sampling the random walk on continuous surfaces. In this work, however, as the space is combinatorial, it was necessary to modify the random walk to deal with combinatorial spaces. The modification consists of selecting an initial solution and traversing the graph advancing through the neighborhood of the last visited solution. In the original work, the random walk consisted of selecting points that are *step*-distant from each other, where *step* is a constant defined according to the size of the continuous space.

2.2.4 Neutrality Rate

Neutrality is a metric designed to measure and identify the presence of regions of the search space with small or no variations in values of fitness, i.e., immediate neighbors with equal fitness. Flat regions in the search space can be a problem for algorithms that are gradient-driven or rely on a local search, such as Hill-Climbing. By definition, neutrality is the opposite of roughness, but both metrics are useful since part of the search

space can present high roughness while other regions can present high neutrality.

Neutrality can both make the search space easier to explore [52] or get some algorithms stuck in regions of the search space with equal (or almost equal) fitness, preventing them from exploring areas with possibly better results [16]. Assuming a discrete representation of the solutions a_g and defining a "mutation" as a change in one of the components of a_g that leads to a neighbor solution $a_g^i \in N(a_g)$, We evaluate the neutrality of our landscape based on neutral walks (as defined by [37]), which performs a random walk and identifies the number of neighbors with fitness lower than the parameter δ.

However, it is important to mention that there other ways to look at neutrality. For example, [51] define a Neutral Neighborhood $N_n(a)$ of a solution a as all the nodes in the neighborhood of a with a sufficiently similar fitness value to $f(a)$, formally: $N_n(a) = \{a' \in \mathcal{N}(a) \mid |f(a') - f(a)| < \epsilon\}$ for a sufficiently small $\epsilon \geq 0$. A Neutral Network is defined as a connected component (subset $\mathcal{S}'$ of the search space $\mathcal{S}$) of Neutral Neighborhoods, formally defined as: $N = (\mathcal{S}', E_{\mathcal{N}})$, where $E_{\mathcal{N}} = \{s_1, s_2 \in \mathcal{S}^2 | s_2 \in N_n(s_1)\}$ [51]. In a neutral network, the Neutrality Degree $N_d(s)$ and Ratio $N_r(s)$ of a solution s are defined as: $N_d(s) = |N_n(s)|$ and $N_r(s) = \frac{|N_n(s)|}{|N(s)|}$. It has been shown that large values (≈ 1) for the Average Neutrality Ratio in a Neutral Network $\overline{N_r} = \frac{\sum_i^n N_r(s_i)}{|\mathcal{S}'|}$ indicate that there is a large number of possible neutral mutations between individuals in a search space [51], and therefore we use this value as one of the metrics for neutrality. In its original formulation, the previously described approach of Neutral Networks requires a full enumeration of the search space. This was done in [24], and will be considered in future works.

Finally, recall that problems with many neutral regions can be difficult for algorithms that use the gradient to guide themselves, since in neutral regions the gradient ($\nabla(\mathcal{F}(\hat{v}))$) can tend to zero in all directions.

2.3 Local Optima Networks

In order to deal with fitness landscapes that have a number of solutions prohibitive to enumerate and aiming to provide a more global view of the space, [28] proposed a new way to analyze and visualize search spaces, named Local Optima Networks (LONs).

LONs look at the fitness landscape of a problem using a graph $G = (V, E)$, where each configuration v of the solution space represents a node in the graph, i.e., $v \in V$, and there is a directed edge $e = (u, v)$ if $v \in \mathcal{N}(u)$ and $\{u, v\} \in V$. The weight w_{uv} of an edge can represent, for example, a distance between the two nodes or the intensity of the relationship between them, and depends on how the operator $\mathcal{N}$ is defined.

In the LON, given the graph G, we need to filter the nodes so that only the

local optima remain. Hence, we can define it as LON $= (V', E')$ where $V' = \{v \,|\, v \in V$ and $\mathcal{F}(v) \geq \mathcal{F}(\mathcal{N}(u))\}$ and $\mathcal{F} : S \to \mathbb{R}$ is a function that maps each solution in the search space to a quality metric, in our case, the fitness. Building the LON highly depends on the concept of node neighborhood, $\mathcal{N}(u)$, as it affects the concept of neutrality and roughness.

While the nodes of a LON are local optima, the edges represent relationships between them, such as their probability of transition [28, 56]. Different ways to assign weights to edges have been previously proposed in the literature:

- **Basin-transition edges**: as the local search defines a mapping from the search space to a set of locally optimal solutions, it also generates basins of attraction. The basin of attraction b_i of a local optimum LO_i is composed by all solutions s in the search space that satisfy $LS(s) = LO_i$, that is, $b_i = \{v \in V(G) \,|\, LS(v) = LO_i\}$. Therefore, in the basin transition method [28], the weight of the edge that connects two local optima LO_i and LO_j is given by:

$$w(e_{ij}) = \frac{1}{|b_i|} \sum_{s \in b_i} \sum_{s' \in b_j} p(s \to s') \tag{2.4}$$

  where $p(s \to s')$ is the probability of a mutation in s generates s'.

- **Escape edges**: given a function $d(s, s')$ that determines the minimum number of edges between two solutions, an escape edge defines the weight of edge e_{ij} connecting LO_i and LO_j as follows [28]:

$$w(e_{ij}) = \frac{|\{u \in V(G) \,|\, d(u, LO_i) \leq D \text{ and } LS(u) = LO_j\}|}{|\{u \in V(G) \,|\, d(u, LO_i) \leq D\}} \tag{2.5}$$

  where D is a user-defined parameter (usually set to 1 or 2, according to [28]).

- **Perturbation edges**: defines the weight of the edge e_{ij} that connects the local optima LO_i and LO_j as the estimated number of times that, after a perturbation/mutation of LO_i followed by a local search, LO_j is generated [2], i.e. ,

$$w(e_{ij}) = \frac{|\{LS(mutation(LO_i)) = LO_j\}|}{n\_trials} \tag{2.6}$$

  where $mutation(LO_i)$ represents the mutation operator being applied to LO_i and n_{trials} is the number of perturbations generated. The value of n_{trials} was set as the number of local optima so that it was possible to verify if the mutation probability is approximately uniformly distributed, i.e., capable of reaching all other solutions in the space.

Having a LON, we can extract from it a set of characteristics – translated as metrics – that come from the complex network literature [1] and can help understand the characteristics of the search space. A detailed explanation of these metrics and how they relate to the characteristics of the space are later discussed in Section 4.1.

Apart from the original LON model [28], two other variations of LONs - namely Monotonic Local Optima Network (MLON) and Compressed Monotonic Local Optima Network (CMLON) [27] are of interest of this work. The original version of LON is known for not supporting search spaces with neutrality (i.e., which the presence of plateaus). Compared to the original model, MLONs discard edges that connect solutions where the quality of the target node is smaller than the quality in the source node. CMLONs, in turn, compress the search space generated by MLONs by collapsing neighbors with the same fitness value into a single local optima. These models will be further discussed in Chapter 4.

2.4 Related work

There are a few studies in the literature that have looked at the landscape of AutoML problems in general. The works performed so far can be divided into two groups: those where the search space is made of machine learning pipelines and those where the space refers to neural network architectures. There are also a number of papers that look at the fitness landscape of neural network loss functions [48], but these we consider interesting but out of the scope of this thesis. However, it is important to notice that a few works have looked at the fitness landscape of the broader problem of algorithm configuration [33].

Going back to AutoML landscape, we start by looking at problems where candidate solutions are full pipelines. [8] were the first to perform the analysis of AutoML landscapes considering a subspace of TPOT. Their objective was to identify the local characteristics of the space close to optimal solutions by using metrics such as slope, roughness and neutrality. Their results suggest that many regions of high fitness exist in the space, but these are prone to overfitting. In this same direction, [32] looked at fitness landscape metrics to better understand the search space of a huge space of machine learning pipelines. They looked at FDC and metrics of neutrality, and concluded FDC was a poor metric for performing the analyses.

Concerning the landscape of algorithm configuration problems, [33] evaluated them in terms of modality and convexity of parameter responses. The authors defined parameter response slices by varying parameter p within a given window around an optima found

by Sequential Model-based Algorithm Configuration (SMAC) [11], keeping all other parameters fixed and measuring the performance of the algorithm as a function of p. They evaluated algorithms for typical optimization problems, and concluded that many of the parameter slices appear to be uni-modal and convex, both on instance sets and on individual instances. Following this work, [34] tested the unimodality of the AutoML loss landscape considering the joint interaction of the hyperparameters and concluded that most landscapes have this property, but are not convex. They also observed that hyperparameters interact strongly in regions of configuration space farther from the optimal solutions. Finally, they also empirically demonstrate that FDC has limitations in characterizing certain spaces, which highlights the need for new methods with LON.

Turning to analysis of spaces of neural network architectures, [24] analyzed the fitness landscape of NAS in the context of graph neural network architectures. They used FDC together with the dispersion metric (which measures the dispersion between the funnels in the landscape), and have also looked at the neutrality of the space. The analysis of neutrality indicated that the space was not neutral, but the authors highlighted the need to use more elaborate techniques for estimating neutrality.

Going further, [46] introduced the concept of fitness landscape footprint, given by an aggregation of eight general-purpose metrics to synthesize the landscape of an neural network architecture search problem. The metrics considered include FDC, space roughness, number of local optima and persistence — a metric defined by the probability of a configuration ranked by a function $Ranking(\cdot)$ keep its initial position when compared to its place in the rank at instant t_0, in their analyzes. They looked at classical image classification benchmark, and concluded the technique allowed them to characterize the relative difficulty to the problem, and that the insights provided may be used to assess the expected performance of a search strategy in each dataset.

Note that all the works reviews so far focus on using standard metrics of FLA, and they show it is difficult to actually use these metrics in such complex landscapes. The disagreements in terms of FDC, roughness and neutrality ask techniques able to provide a more global view of the landscape, and this is where Local Optima Networks (LONs) become important.

[47] adapted LONs to analyze the global structure of parameter configuration spaces. They looked at the metrics extracted from LONs and FDC, and observed large differences when tuning the same algorithm for different problem instances. For complex scenarios, they found a large number of sub-optimal funnels, while simpler problems had a single global funnel. With this same objective, the authors in [3] looked at parameter spaces for Particle Swarm Intelligence (PSO), and found that PSO's parameter landscapes are relatively simple at the macro level but a lot more complex at the micro level, making parameter tuning more difficult than they initially assumed.

Finally, a few recent works have looked at the fitness landscapes of NAS prob-

lems from a point of view of both classical metrics of FLA and LONs. [40] analyzed the NAS fitness landscape generating a fully enumerable space using FDC and LON. The results showed that the search space is easy and that most local optima are only one perturbation away from the global optimum. They conclude that it is possible to apply tools to characterize search landscapes to small problem instances, and use the obtained information to effectively infer properties of the larger search space. In a previous work, [41] used autocorrelation and entropic measure of ruggedness to characterize the performance of neuroevolution of convolutional neural networks. The results were obtained on four different test problems, and confirm that these measures are reasonable indicators of problem hardness, both on the training set and on the test set.

[26] have also looked at fitness landscapes of NAS benchmarks. They concluded that the FL analyzed are multimodal but have few local optima, making it not complicated for local search methods to escape these regions of the search space with simple perturbation operations.

Chapter 3

Exploratory Landscape Analysis of AutoML Search Spaces

This chapter defines the problem of fitness landscape within the scope of AutoML problems, and addresses the problem posed in RQ1: *Are traditional metrics of fitness landscape analysis appropriate to characterize and indirectly measure the difficulty of AutoML problems?*

We first define the fitness landscape, and then characterize it over a set of 20 datasets. We then applied the metrics discussed in Section 2.2. In addition to presenting the results, we discuss the effect of different representation types and distance metrics on the result.

3.1 Proposed AutoML Fitness Landscape

Given the formal definition of the AutoML problem and its fitness landscape introduced in Section 2.1, we first need to define the set of algorithms and hyperparameters that can compose the space of pipelines, and then define the neighborhood and fitness function to create the fitness landscape.

The types of AutoML fitness landscapes previously analyzed in the literature were either oversimplifications of an AutoML search space or too complicated for us to generate the complete space [32]. We generate an AutoML search space using a trade-off: the space has enough components to generate robust solutions to real problems but it is simple enough so it can be fully enumerated.

As previously explained, one of the peculiarities of AutoML spaces is the high number of conditional hyperparameters, i.e., a hyperparameter which only exists if another hyperparameter is previously selected. For example, depending on the classifier selected for the AutoML problem being solved, the set of hyperparameters can be completely different: while AdaBoost has only two parameters, namely, algorithm and number of

estimators, Random Forest has seven. Due to this large number of conditional parameters, we use a grammar to generate feasible solutions. Each solution is generated by using a derivation tree. The grammar allows the produced ML pipeline to choose between a set of preprocessing steps (or none) followed by a classification algorithm, as detailed in Section 3.1.1. The neighborhood is generated by the mutation operator (Section 4.1.1), and the fitness is based on the f-measure of the solutions (Section 3.1.3).

3.1.1 Search Space

We use a grammar to generate the search space of AutoML pipelines, and each solution comes from a derivation tree extracted from the grammar. The grammar has 38 production rules, 92 terminals and 45 non-terminals, and is available in Appendix A. In terms of preprocessing, it includes algorithms that deal with feature scaling and dimensionality reduction, such as Principal Component Analysis (PCA) and Select K-Best. It is also possible for a pipeline to use no preprocessing algorithms. In terms of classification methods, there are five possible options: Logistic Regression, Multilayer Perceptron, K-Nearest Neighbors (KNN), Random Forest, and Ada Boost. The number of hyperparameters varies according to the selected algorithm, going from two (Ada Boost) to 7 (Random Forest).

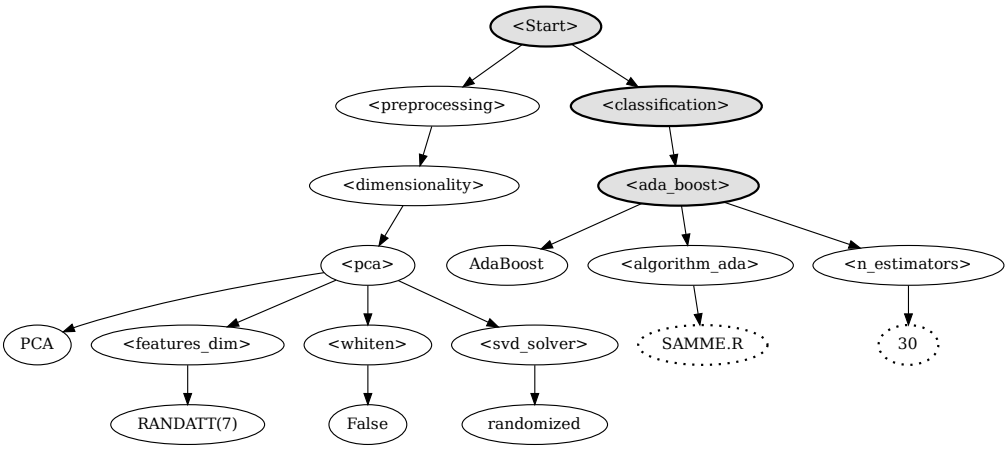
A few continuous parameters are left out of the grammar, and assume their default values as defined in their Scikit-learn implementation [31]. For parameters that depend on some external attribute, such as the dataset dimension for feature selection and dimensionality reduction, for example, relative values (percentages) were used to avoid generating invalid numbers when calculating the suitability of pipelines to different datasets.

In order to give the reader an idea of the impact of conditional parameters, if non-terminal combinations of the grammar are considered without any constraints, the generated space has 221,562 possible solutions. When conditional parameters are added, the search space size is reduced to 69,960 (30% of the non-constrained space). Note that the grammar was generated in a way that its solutions can be completely enumerable. This is essential to validate the FLA metrics.

3.1.2 Solution Representation and Neighborhood

Given the search space is defined by a grammar, the most intuitive way to represent the candidate solutions is by using the derivation tree directly extracted from the grammar. Figure 3.1 shows an example of a pipeline produced from the grammar, which uses PCA as preprocessing algorithm and AdaBoost as classifier. It is an example of a derivation tree extracted from the grammar. Note that the algorithms and hyperparameters are the leaf nodes of the tree.

Figure 3.1: Example of AutoML pipeline.



Source: Elaborated by the author.

Given a representation, we also need to define the concept of solution neighborhood, which is usually defined from a distance between pairs of solutions. In the case of machine learning pipelines, we do not have an inherent concept of distance/similarity that can easily determine how distant they are. For example, given two ML pipelines with the same preprocessing and different classifier, should they be considered closer, more distant or as distant as two pipelines with different preprocessing but the same classifier?

Given this drawback, [32] introduced an *ad hoc* technique to define the distance between AutoML pipelines represented by grammar-derivation trees, where the distance between individuals with different algorithms is greater than that of individuals that differ only in hyperparameters. [24], in turn, measures the distance between pipelines by first converting them to a binary representation using *one-hot* encoding and then calculate the Hamming distance between these binary representations. In addition, the authors also apply the t-SNE algorithm [49], used for dimensionality reduction, to generate a dense space and use the Euclidean distance to calculate the distance between the pipelines in the embedded space.

Table 3.1: Pipeline node classification. These nodes are used to generate the distances between different groups using expert knowledge.

A0: `NULL` symbol	A9: Neural networks
A1: `<start>` symbol	A10: Nearest neighbors
A2: `<preprocessing>` symbol	A11: Discriminant analysis
A3: `<classification>`	A12: Trees
A4: Imputation algorithms	A13: Ensembles
A5: Data range manipulation algorithms	A14: Discrete hyperparameters
A6: Dimensionality manipulation algorithms	A15: Continuous hyperparameters
A7: Naïve Bayes	A16: Categorical hyperparameters
A8: Linear models	

Source: Elaborated by the author.

As the results of FLA metrics are directly influenced by the concept of distance adopted by the authors, we adopted and compared three different representation and distance methods according to previous work: (i) we used the method proposed by [32], based on tree representations, from now on referred as *ad hoc*, (ii) we used the methods based on binary representation (*one-hot* encoding) and iii) the embedded space generated by t-SNE, both proposed by [24] in the context of understand the fitness space of graph neural networks. Next, we discuss each of these methods.

***ad hoc* distance method in tree-based representation:** This method assigns constant values to represent the distance between each type of node present in the pipeline. 16 types of nodes are defined, as listed in Table 3.1, and then their distances are calculated according to the values defined in Table 3.2, assigned by specialists.

The final distance between two pipelines is equal to the sum of the distances of the nodes contained in each one. The idea of this method is to consider that the impact of changing an algorithm is more significant than changing the value of a hyperparameter and that the presence of an algorithm is greater than simply changing the algorithm, as is the case with distance between a preprocessing algorithm and an empty node, where $d(A2, A0) = d(A0, A2) = 8$.

Hamming distance in binary representation: The second method to define the distances between pairs of pipelines first converts the tree into a binary sequence using *one-hot* encoding, which is a simple way of transforming structured or categorical data into a numerical representation. The Hamming distance is then used to calculate distances between solutions.

The transformation to *one-hot* encoding is done as follows. Each pipeline P is represented by a binary sequence $\mathcal{S}^{\mathcal{P}}$, where the presence or absence of a terminal corresponds to 1 or 0, respectively. Each terminal is assigned a specific and fixed position in the sequence, so all pipelines composed of the same terminal have 1 in the position representing that terminal. For example, if the classification algorithm X is mapped to the i-th position of the sequence $\mathcal{S}$ and a given pipeline P has by this algorithm, then the sequence representing that pipeline has the i-th element equals to 1, i.e., $\mathcal{S}_i^P = 1$.

Table 3.2: Distances between pairs of node types.

	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
A0	1	0	8	0	4	4	4	0	0	0	0	0	0	0	0	0	0
A1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A2	8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A4	4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
A5	4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
A6	4	0	0	0	0	0	1	2	2	2	2	2	2	2	0	0	0
A7	0	0	0	0	0	0	2	1	2	2	2	2	2	2	0	0	0
A8	0	0	0	0	0	0	2	2	1	2	2	2	2	2	0	0	0
A9	0	0	0	0	0	0	2	2	2	1	2	2	2	2	0	0	0
A10	0	0	0	0	0	0	2	2	2	2	1	2	2	2	0	0	0
A11	0	0	0	0	0	0	2	2	2	2	2	1	2	2	0	0	0
A12	0	0	0	0	0	0	2	2	2	2	2	2	1	2	0	0	0
A13	0	0	0	0	0	0	2	2	2	2	2	2	2	1	0	0	0
A14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0.5
A15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0.5
A16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0.5

Source: Elaborated by the author.

It is important to say that the one-hot encoding process was lexical, i.e., two different parameters that were allowed the same value in different contexts were assigned the same position in the vector. For example, the number 5 meaning trees of a Random Forest or neighbors in a KNN were mapped to the same place. Hence, instead of having a length of 96 (the number of terminals of the grammar), the *one-hot* encoding has length 64.

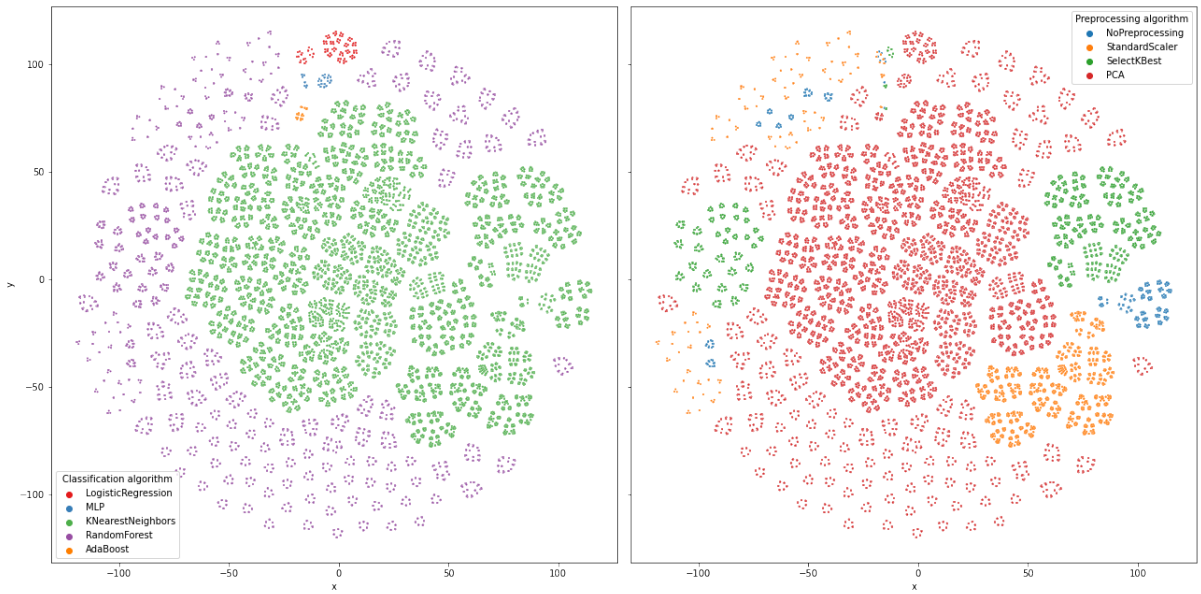
Euclidean distance in 2D embedded representation: The third method of distance uses the (sparse) representation generated by the *one-hot* encoding with high dimensionality and condenses the vector into a two-dimensional space $\mathbb{R}^2$ using the t-SNE algorithm, run with the default parameters values of its Sklearn implementation. These values are then compared using the Euclidean distance between the two representations.

Reducing the representation to a $\mathbb{R}^2$ space is interesting because it allows the visualization of the configuration space, as shown in Figure 3.2. In the figure on the left, the color indicates the classification algorithm used in the pipeline. In the figure on the right, the color indicates the preprocessing algorithm used. Observe that the distribution of pipelines using a specific algorithm is not uniform and this occurs because certain algorithms have more hyperparameters than others.

3.1.3 Fitness Function

Having the solution space defined, next we define the fitness of the solutions was defined as the weighted average F-measure [58] (Equation 3.1):

Figure 3.2: Dimensional reduction of the generated space using *one-hot* encoding. The original space has 64 dimensions, here represented as two.



Source: Elaborated by the author.

$$\text{F-measure} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{3.1}$$

which is the harmonic mean of precision and recall. In the equation, TP stands for the true positives, FP for the false positives, and FN for the false negatives. As some datasets have several classes, the one-vs-all strategy was employed when calculating this metric.

When evaluating the fitness, a maximum computational budget (for training and testing the pipeline) was defined, and solutions that exceeded this limit received fitness 0. This was the naive way we found to deal with the trade-off between the computational cost versus quality of the solutions.

3.2 Characterization of the Fitness Landscapes

The fitness landscape of a problem depends directly on the data being analyzed. In this work the pipelines were evaluated in 20 datasets selected from UCI Machine Learning Repository[1] and from Kaggle[2]. The selection criteria were: (i) popularity, (ii) numerical or categorical features and (iii) the task intended was classification.

[1]https://archive.ics.uci.edu/ml/index.php
[2]https://www.kaggle.com/datasets

Considering the search space defined in Section 3.1, we generate all the solutions and evaluated them for each of the 20 datasets, generating 20 different fitness landscapes. Table 3.3 presents some features of the datasets used to generate the fitness landscape. The "Code" column indicates the code used to reference each dataset, the "Instances" column indicates the number of instance, the "Features" column indicates the number of features, the the "Classes" column indicates the number of classes present in the target feature. Following, the "Optimum" column indicates the fitness of the global optimum (from the space defined by the grammar) and the "#Optimum" column indicates the number of solutions that achieve the value of optimal fitness.

Table 3.3: Characterization of the datasets.

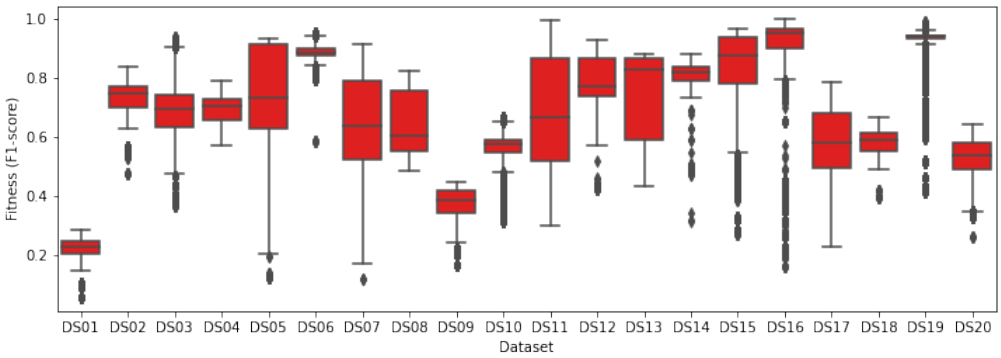
Dataset	Code	Instances	Features	Classes	Optimun	#Optimum
abalone	DS01	4177	8	28	0.2842	1
bank	DS02	11162	16	2	0.8376	8
car-evaluation	DS03	1728	6	4	0.9380	8
diabetes	DS04	768	8	2	0.7900	8
dry-bean	DS05	13611	16	7	0.9309	32
fire	DS06	17442	6	2	0.9539	8
fruit	DS07	898	34	7	0.9157	1
heart	DS08	303	13	2	0.8216	96
ml-prove	DS09	6118	51	6	0.4478	21
mushrooms	DS10	8124	22	7	0.6678	10
nursery	DS11	12960	8	5	0.9937	1
pistachio-28	DS12	2148	28	2	0.9295	6
pumpkin	DS13	2500	12	2	0.8829	3
raisin	DS14	900	7	2	0.8810	1
statlog-segment	DS15	2310	19	7	0.9685	24
texture	DS16	5500	40	11	0.9980	6
vehicle	DS17	846	18	4	0.7875	7
water-potability	DS18	3276	9	2	0.6643	1
wilt	DS19	4839	5	2	0.9888	2
wine-quality-red	DS20	1599	11	6	0.6402	16

Source: Elaborated by the author.

Further, Figure 3.3 shows the box-plots of the fitness distribution of the pipelines generated for each dataset. Note that, for some datasets, the fitness of the solutions is predominantly high or low, while for others they are better distributed. Observe that this distribution does not affect the FLA, but gives an insight on the difficulty of the problem.

Table 3.4 shows more detailed statistics of the fitness of the pipelines evaluated for each dataset, summarized in the box-plots of Figure 3.3. The variance is relatively low since the fitness (F1-score) can vary from 0.00 to 1.00. The column #Duplicates indicates the number of non-unique fitness values and the column μDuplicates indicates the average number of repetitions of the non-unique fitness. For example, if there are 10 solutions with fitness 0.8 and all others have unique fitness values, then #Duplicates is 1

Figure 3.3: Box-plot of the fitness of the pipelines in different datasets.



Source: Elaborated by the author.

and μDuplicates is 10.

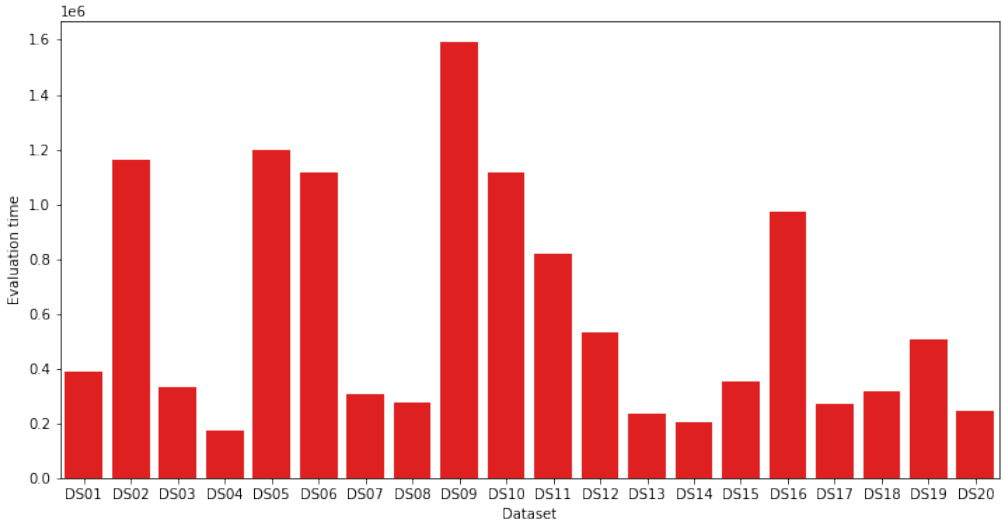
Table 3.4: Fitness of each dataset.

Code	Variance	Median	Mean	Max	Min	#Duplicates	μDuplicates
DS01	0.0009	0.2278	0.2224	0.2842	0.0544	332	210.6898
DS02	0.0077	0.7480	0.7197	0.8376	0.4763	974	71.8090
DS03	0.0091	0.6959	0.6897	0.9380	0.3669	469	149.1599
DS04	0.0022	0.7063	0.6949	0.7900	0.5706	108	647.7778
DS05	0.0229	0.7307	0.7597	0.9309	0.1206	874	80.0240
DS06	0.0007	0.8870	0.8919	0.9539	0.5865	916	76.3526
DS07	0.0194	0.6359	0.6527	0.9157	0.1176	318	219.9151
DS08	0.0100	0.6056	0.6449	0.8216	0.4836	71	985.3521
DS09	0.0021	0.3843	0.3767	0.4478	0.1646	606	115.4406
DS10	0.0080	0.5764	0.5474	0.6678	0.3132	894	78.2494
DS11	0.0313	0.6669	0.6890	0.9937	0.2986	2529	27.6161
DS12	0.0053	0.7706	0.7912	0.9295	0.4222	396	176.6389
DS13	0.0181	0.8280	0.7376	0.8829	0.4309	289	242.0657
DS14	0.0012	0.8206	0.8145	0.8810	0.3127	95	736.3053
DS15	0.0261	0.8782	0.8120	0.9685	0.2678	484	144.5103
DS16	0.0556	0.9537	0.8306	0.9980	0.1603	588	118.9439
DS17	0.0098	0.5784	0.5876	0.7875	0.2260	225	310.9022
DS18	0.0012	0.5898	0.5844	0.6643	0.3928	303	230.8746
DS19	0.0026	0.9421	0.9316	0.9888	0.4109	325	215.2431
DS20	0.0036	0.5384	0.5293	0.6402	0.2607	252	277.6111

Source: Elaborated by the author.

Figure 3.4 shows the total time required for evaluating the entire configuration space. The total time is equal to the sum of the time needed to train and evaluate each solution individually. Some factors that justify the variation in total time are the number of samples and the size of each dataset. For example, DS09 is the largest dataset (51 features). The experiments were run on an Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz and approximately 65GB of RAM.

Figure 3.4: Evaluation time for the complete search space defined by the grammar.



Source: Elaborated by the author.

3.3 Results and Discussion

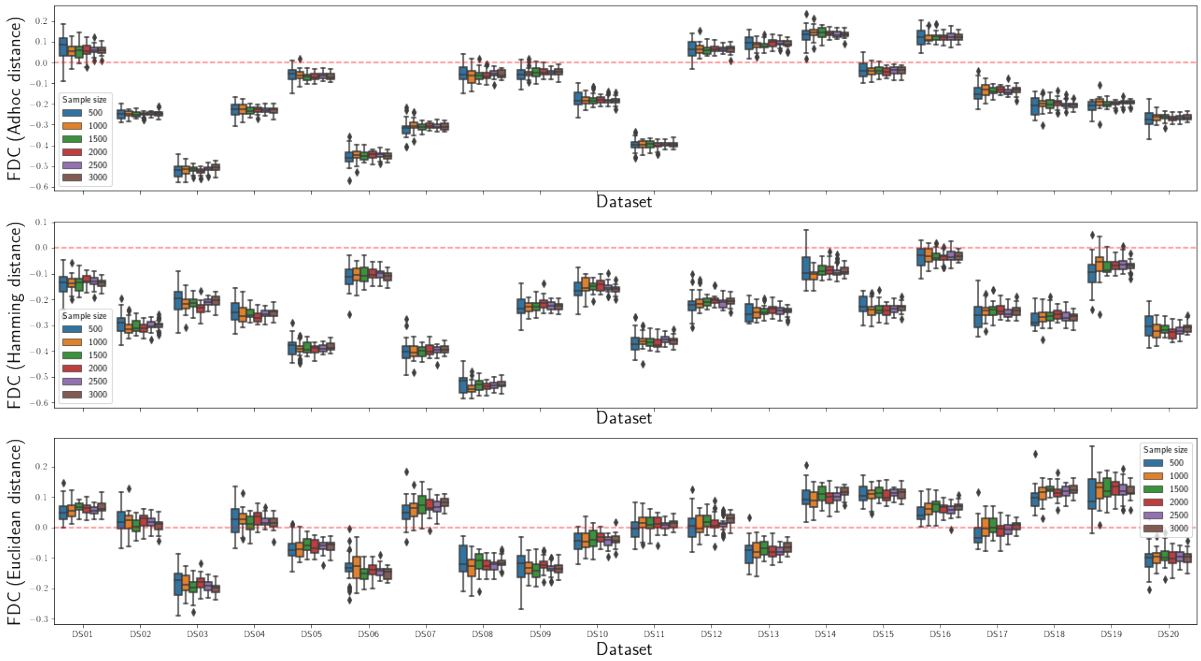
This section presents and discusses the results of the metrics previously discussed in the AutoML fitness landscape defined in Section 3.1.

3.3.1 Fitness Distance Correlation

FDC is a metric commonly used in the AutoML literature that depends directly on the concept of distance. As previously explained, this work considers 3 distance metrics: *ad hoc* distance, hamming distance and euclidean distance. The experiments were performed using a set of 500 to 3,000 solutions with steps of 500 randomly selected in the configuration space. As several datasets have more than one global optimum, the distance used was the smallest, that is, the distance from the global optimum closest to the observed solution. To calculate the co-variance between the distances and the fitness, the function `numpy.cov()` from the *numpy* library was used with the parameter *bias* equal to `True`, which affects the factor in the division from $N-1$ to N, where N is the sample size.

We analyze the values of FDC over two perspectives: in the first, we consider that global optimum is known. In the second, we use as the reference point to the metric the

Figure 3.5: FDC (distance to global optimum).



Source: Elaborated by the author.

closest local optimum. We performed this analysis because, in most scenarios, the global optimum is not known, and FDC ends up looking at the local optima of the sampled space. Instead of sampling the space, we calculated the metric using the closest local optima. We show that the results of this second approach vary substantially according to the analyzed datasets.

Results considering the global optimum: Figure 3.5 shows the box-plots grouped by dataset, and the color indicates the length of the walk. Experiments were performed using a set of six different walks in the interval of 500 to 3,000 solutions, selected randomly from the search space considering intervals of size 500.

Observe that the FDC metric tends to negative values for most datasets, but the correlation is not strong for any of the cases, as the FDC $\leq$ -0.6 in all cases. The fact that the result is negative indicates that as the distance of a given solution contained in the random walk to the closest global optimum increases, the fitness of the solution also increases. The walk length can affect the variance of the FDC as can be seen for the largest bar in the box-plots referring to walks of length equal to 500, however the difference between the FDC with the largest and the smallest walk length is not significant. In the case of Euclidean distance, specifically, the distribution of the FDC is more uniform and 11 instances have a mean greater than zero.

Another point to be considered is that increasing the neighborhood has no effect on the result obtained by the metric, although this is a factor that directly affects the difficulty of the space, as we showed in our work [45]. This is because all methods

measure the distance in relation to the same global optima and because, by definition, the distance metrics used only consider the syntactic characteristics of the pipelines (they do not consider the topological structure of the search space). Therefore, the result obtained with the neighborhood of size 15, 20 and 25 result in the same values. However, if the distance is measured in relation to a local optima, the results are influenced by the characteristics of the space, as the local optima change when the neighborhood changes.

Although in absolute terms the FDC values are different depending on the distance calculation method, the results are expected to maintain the same relationship between the datasets, i.e., if the datasets are ordered according to FDC, the same order should be maintained regardless of the distance adopted. This is important because if the order changes, it indicates that the FDC results are only comparable if the same distance is used in the experiments.

To compare whether the order is in fact maintained, the datasets are ranked according to FDC for each distance measure used, and the rankings are compared using the Kendall coefficient τ [21] to measure the correlation between rankings. This method calculates the number of concordant/discordant pairs that are present in both rankings. In this context, a matching pair means that given two ranks R_1 and R_2 containing N elements sorted independently. If an element e_1 appears before an element e_2 in both rankings, then e_1 and e_2 are said to be matching pairs, even though the positions of e_1 and e_2 are different, and the opposite generated discordant pairs.

If the rankings are composed of the same N elements possibly ordered differently, then there are $\binom{N}{2}$ possible pairs. Considering that C denotes the number of concordant pairs and D the number of discordant pairs, the coefficient τ is defined as $\tau = (C - D)/(C + D)$. If D = 0, then the expression reduces to $C/C = 1$, that is, if all pairs agree, then the coefficient is equal to 1. If C=0, then the resulting expression is $-D/D = -1$, that is, if all pairs are discordant, then $\tau = -1$. Therefore, $\tau \in [-1, +1]$, and the higher the value, the more similar the rankings.

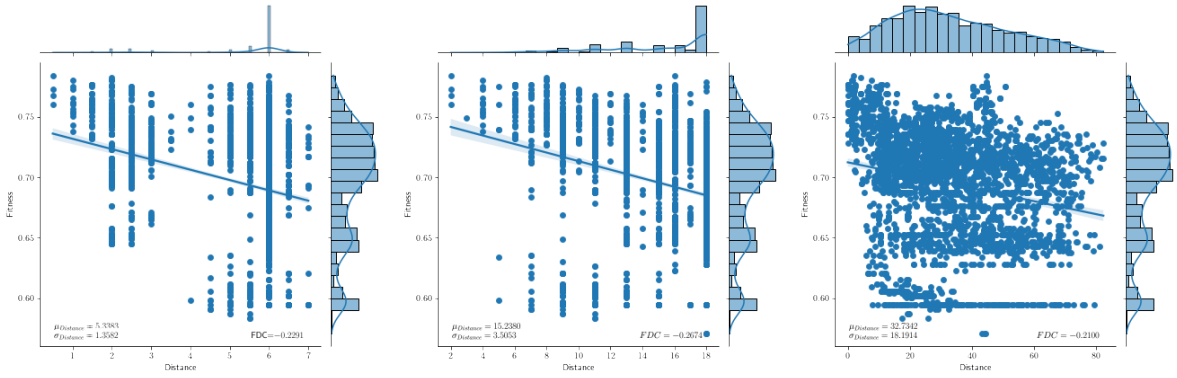
The ideal scenario is that the correlation is positive and close to +1, indicating that the results are robust to a metric that is not strongly established, such as the distance between pipelines. However, the results show, with significance $\alpha = 0.05$, that the correlation between the rankings is low, and hence the representation and distance significantly affect the FDC results using the global optimum as a reference, as indicated below:

$$\begin{array}{ll} \tau(\textit{ad hoc}, \text{hamming}) = 0.4421 & \text{p-value}=0.0983 \\ \tau(\textit{ad hoc}, \text{euclidean}) = 0.5263 & \text{p-value}=0.8227 \\ \tau(\text{hamming}, \text{euclidean}) = 0.3263 & \text{p-value}=0.1126 \end{array}$$

In conclusion, FDC is highly affected by the representation and distance definitions.

An alternative way of analyzing FDC is through a joint plot of the distance by the fitness of each solution contained in the sampled random walk, as depicted in Figures 3.6a-3.6c. Each figure represents a different type of distance and the mean and standard deviation of the distances of each solution contained in the walk are shown in the lower left corner and the FDC in the lower right corner. The *ad hoc* and hamming method concentrate the distance in a specific range while the euclidean distance has a standard deviation of approximately half of the mean.

Figure 3.6: Scatter plot of FDC calculated on DS04 (distance to global optimum)



(a) *ad hoc* distance (b) Hamming distance (c) Euclidean distance

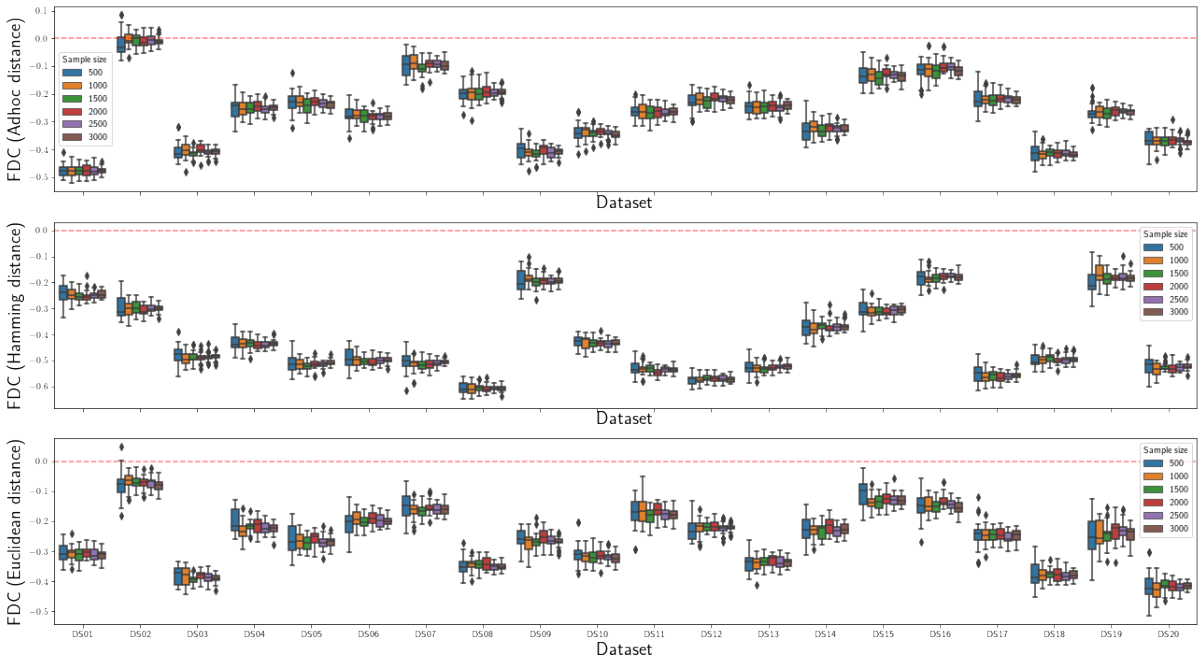
Source: Elaborated by the author.

Results considering the nearest local optimum: Next, we analyze FDC using the distance to the nearest local optima instead of the global optimum, and observe a reduction of FDC values for all distances. In Figure 3.7 it is possible to observe that the FDC calculated using the *ad hoc* and hamming distances have negative values regardless of the size of the adopted random walk, while for the euclidean distance FDC remains relatively close to zero. In all cases, the value of FDC is greater using the distance from the global optimum than from the local optimum. The fact that the FDC is negative indicates that moving away from the local optimum leads to regions with higher fitness, that is, there are solutions with high fitness close to optimal locations with low fitness.

Moreover, increasing the neighborhood changes the set of local optima present in the configuration space and this influences FDC. The impact of increasing the neighborhood affects each dataset differently. For example, the FDC of the instances evaluated in DS01 and DS02 using the euclidean distance are -0.0964 and -0.1816, respectively, with the neighborhood size 15. Increasing neighborhood size to 20 generates an FDC of -0.1104 and -0.1690, respectively, which corresponds to an increase in the absolute value of approximately 15% for DS01 and a decrease of 7% DS2. Therefore, when the distance has the nearest local optima as a reference, the FDC becomes sensitive to the relationships between the solutions contained in space.

Finally, we use the F-ANOVA statistical to verify whether the results are, from

Figure 3.7: FDC of 20 datasets with 15 neighbors. The color of the box-plots represents different walk lengths and each plot refers to a distance metric.



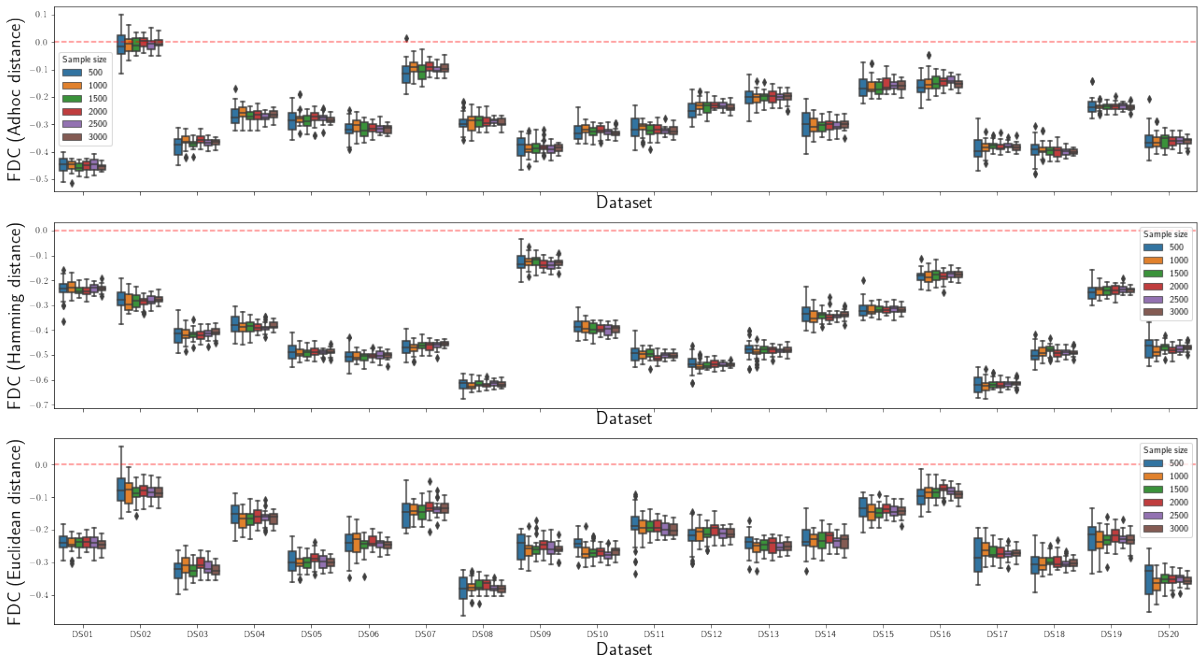
Source: Elaborated by the author.

a statistical point of view, significantly different from each other. To verify this, an experiment was carried out comparing samples with the same neighborhood size and the same walk length, that is, each sample represents the FDC of a particular instance of the problem. The results show that the largest p-value is on the order of 10E-15, that is, the null hypothesis, H_0, can be rejected in all cases. In short, it shows that the distance statistically affects the results of FDC both in relation to the global optimum or the nearest local optimum.

Figures 3.10a-3.10c presents the FDC of DS04, which contrasts with those of Figure 3.6. When the distance is measured in relation to the local optimum, the *ad hoc* distance has a smoother distribution, centered on $d = 1.5$. The FDC using the hamming distance also suffered changes in the distance distribution of the solutions on the random walk path, as it is possible to observe through the concentration of solutions close to $d = 6$. As for the euclidean distance, the solutions are closer to $d = 0$ and decay smoothly as the distance increases.

FDC with global vs local optima Figure 3.11 shows the correlation between the values of FDC obtained when using the distance to the nearest global optimum (indicated by suffix ".g" in the labels of the figure) versus the distance to the nearest local optima (indicated by suffix ".l"). Observe that, in most cases, the correlation is small ($|corr| \leq 0.5$). For example, the metric versions that used the global distance, with the exception of the hamming distance, have a low correlation with all other metrics. In the case of FDC using the hamming distance, the correlation between the global and local versions

Figure 3.8: FDC of 20 datasets with 20 neighbors using *ad hoc* distance. The color of the box-plots represents different walk lengths.



Source: Elaborated by the author.

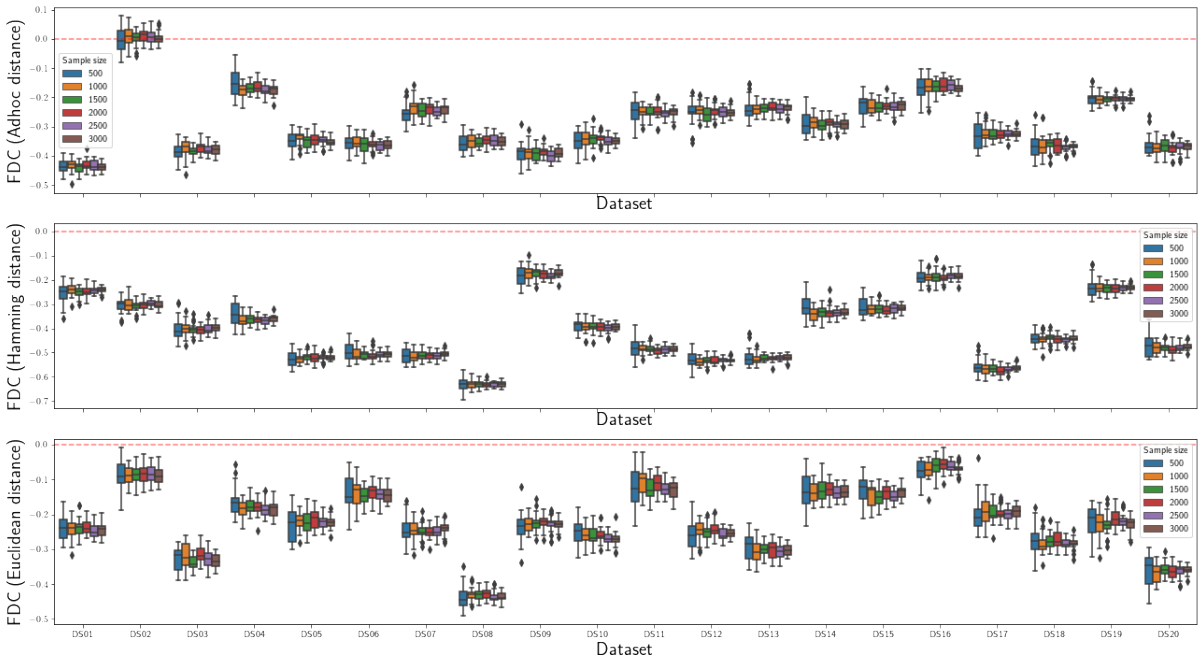
is moderately higher, but it is not considered a strong correlation ($|corr| \geq 0.7$). On the other hand, the correlation is high between metrics that use the same distance from the nearest local optima, but differ only in the size of the neighborhood (indicated by the number in the labels of the figure). Also note a strong Pearson correlation between *ad hoc* and euclidean distances, specially for smaller neighborhoods (Figure 3.11a).

3.3.2 Dispersion Metric

The DM measure gives an indication of how solutions with top-N fitness are distributed in the search space. The experiments were performed using samples of 1,000 and 5,000 solutions with thresholds of 0.01, 0.05, and 0.1. Figure 3.12 presents the value of the metric measured with the largest sample size, i.e., containing 1,000 solutions. Each group is formed by an instance and each bar represents the result of the metric for a different threshold. Similar to the previous experiments, the DM was also repeated 30 times and the error lines indicate the 95% confidence interval.

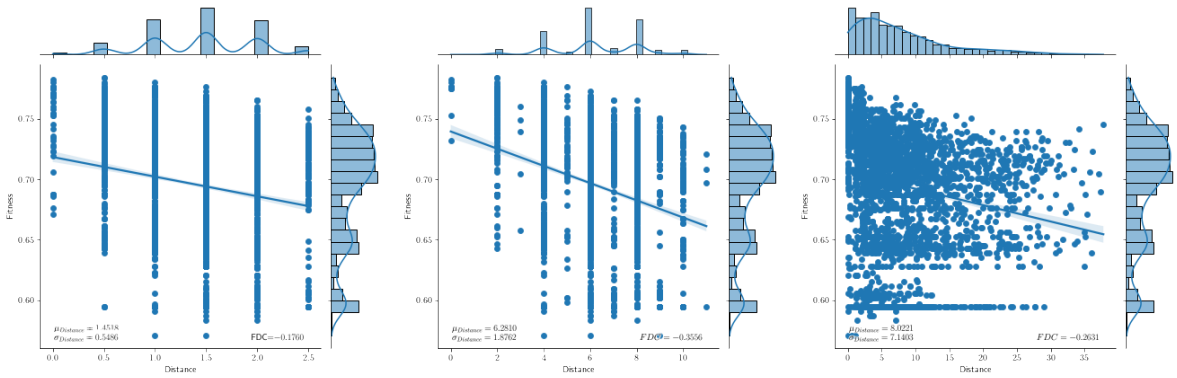
The value of this metric represents the difference between the average of the distances between the top-N solutions, $S^*_{\mathcal{F}}$, with the other solutions $S_{\mathcal{F}}$ in the solution space. A large negative value indicates that the distance between the solutions in $S^*_{\mathcal{F}}$ are closer

Figure 3.9: FDC of 20 datasets with 25 neighbors using *ad hoc* distance. The color of the box-plots represents different walk lengths.



Source: Elaborated by the author.

Figure 3.10: FDC when considering the nearest local optimum.



(a) *ad hoc* distance (b) Hamming distance (c) Euclidean distance

Source: Elaborated by the author.

(concentrated) in the space than the others. However, observe that, for some datasets, the results change completely depending on the distance metric used: the metric goes from a positive to a negative value only by varying the way the distance is calculated, indicating that dispersion, as well as FDC, is highly influenced by the adopted solution representation.

Figure 3.12 shows the values of DM using the three different measure of distances. Note that, when using the *ad hoc* or euclidean distances, the values are relatively distributed between positive and negative. However, when the hamming distance is applied (Figure 3.12b), the metric value is predominantly negative. In all cases the magni-

tude of the metric varies a lot: in the *ad hoc* distance the value varies in the range $(-2.8799, 1.2737)$, in the hamming distance the variation range is $(-10.1791, 0.0646)$ and in the euclidean distance the range is $(-41.7089, 24.5236)$.

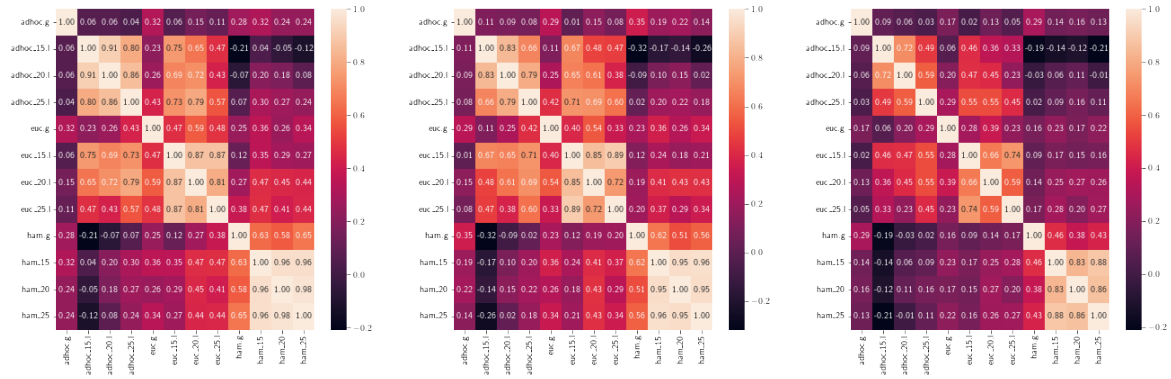
DS10 presents the highest value in the *ad hoc* and euclidean methods, indicating that at this distance the solutions with the highest fitness are "dispersed" in the solution space (according to the distance metric). In the case of the hamming distance, DS11 presents the highest value for DM. This indicates that even receiving the same input data, the results vary completely due to the choice of distance.

A statistical test (ANOVA) was used to verify the difference between the means of the three distances used. For each threshold, a test was performed considering 3 samples, one for each distance calculation method. For example, for threshold 0.01, 3 samples were tested, each containing 20 values (DM for the fitness landscape of each dataset). The results show that the p-value obtained in each test was 0.06226, 1.2723×10^{-6} and 9.8619×10^{-10} for the thresholds 0.01, 0.05 and 0.10, respectively. Thus, considering a sensitivity of 0.05, the null hypothesis that the samples have the same mean can be rejected. Therefore, it is possible to conclude that the results obtained are statistically different due to the choice of the distances and representations.

3.3.3 First Entropic Metric (FEM)

FEM can measure the roughness, neutrality or smoothness of the fitness landscape. We performed experiments using random walks of size from 1,000 to 10,000 with steps of

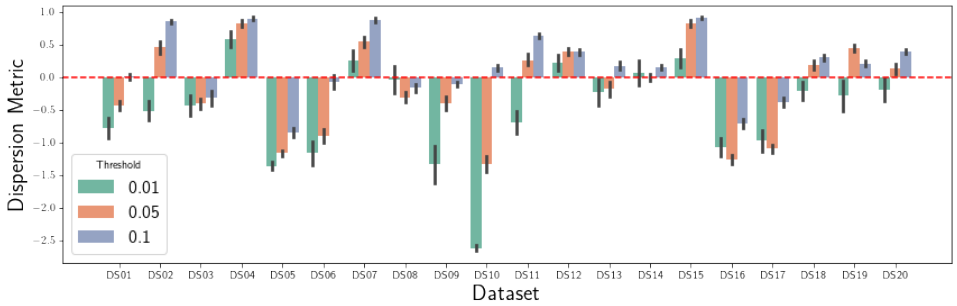
Figure 3.11: Correlation between the FDC calculated with different optima and distance measures.



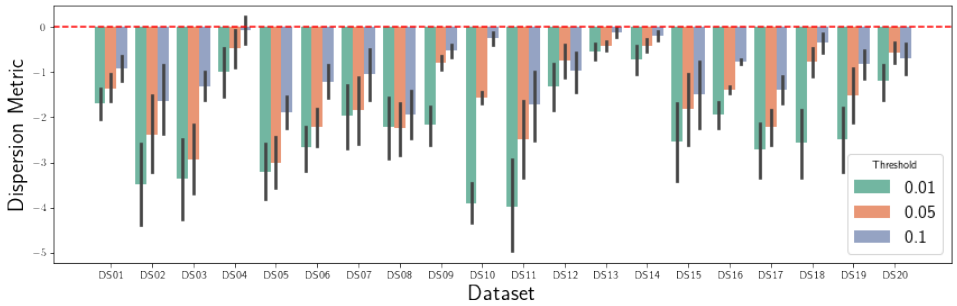
(a) Pearson correlation (b) Spearman correlation (c) Kendall correlation

Source: Elaborated by the author.

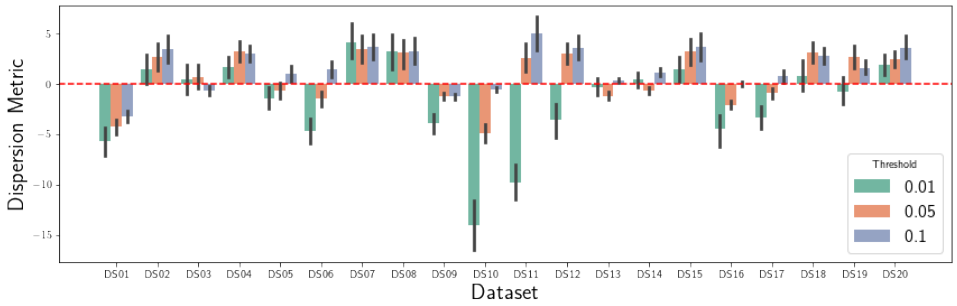
Figure 3.12: Results of the dispersion metric (DM).



(a) *ad hoc* distance



(b) Hamming distance



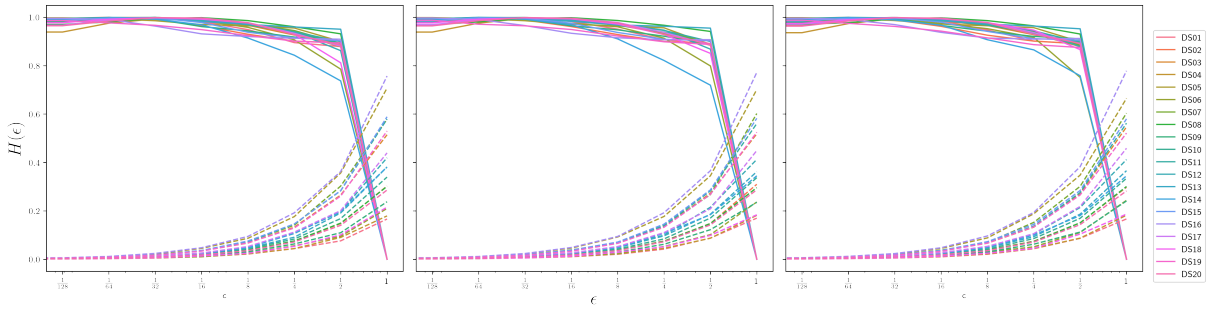
(c) Euclidean distance

Source: Elaborated by the author.

1,000. The results presented were obtained with 30 independent runs. The parameter ϵ^* was defined as the largest difference in fitness between connecting points resulting from the sampling using the random walk. The values of the constant that multiplies $\epsilon*$ were 0, 1/128, 1/64 , 1/32, 1/16, 1/8, 1/4, 1/2, 1, as per experiments carried out in [17].

Figure 3.13 shows the entropy for various values of ϵ. Solid lines indicate the value of FEM in each dataset and dashed lines of the same color indicate the value of $\epsilon*$. This metric was originally developed for continuous landscapes and adapted in this work for a combinatorial landscape. Observing the graphs and considering the results reported in [17], the roughness of the studied space is greater here. This is because, even though the pipelines are neighbors, changing one or more hyperparameters can generate a non-continuous space. For example, some ML algorithms have hyperparameters that represent

Figure 3.13: Neighborhood FEM of size 15, 20 and 25, respectively.



Source: Elaborated by the author.

other algorithms, as is the case with ensemble methods. Because of that, the obtained results are not useful for analyzing the fitness landscape of combinatorial problems.

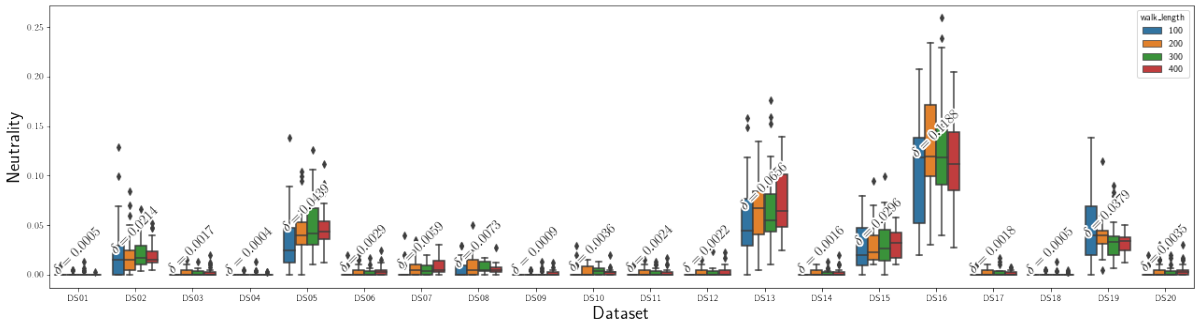
3.3.4 Neutrality Rate

Finally, we measured the neutrality rate of the space using random walks containing 100, 200, 300, and 400 solutions. The value of δ, the tolerance for considering whether a configuration is neutral, was defined as the standard deviation of the fitness mean of 30 random walks of size 1,000, as used in the experiments performed by [32].

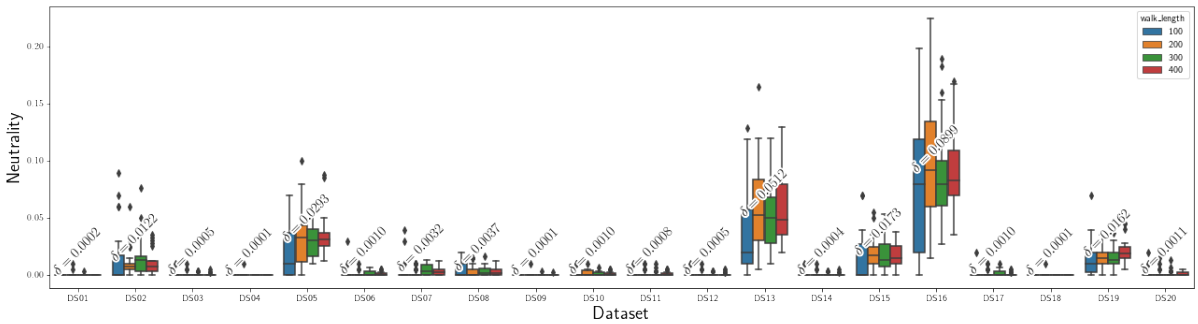
The results are shown in Figure 3.14. Observe that datasets DS16, DS13 and DS05 stand out for having greater neutrality than the others for all walk lengths. However, the dataset with the highest number of repeated fitness, on average, is DS08 – where each of the 71 non-unique fitness occurs approximately 985 times. Note that the number of repeated fitness values does not directly imply space neutrality, since the solutions may not be neighbors. From the 3 datasets with highest values of neutrality, DS16 ranks 14 in terms of repeated solutions. Another factor that justifies this result is the fact that DS16 has the highest fitness variance, which affects the tolerance to consider the neutrality of the neighborhood.

Also notice that increasing the neighborhood size affects the neutrality of space, as can be seen in Figures 3.14b and 3.14c. This is intuitive, as the more neighbors the higher the probability that one has a fitness greater than δ. In addition, note that datasets DS13, DS05, and DS02 are the ones that presented the highest neutrality according to this metric.

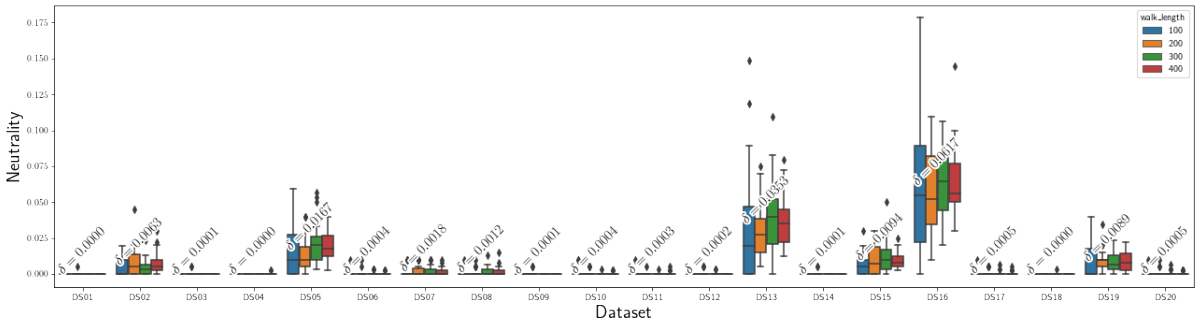
Figure 3.14: Space neutrality measured with different walk lengths.



(a) $|\mathcal{N}| = 15$



(b) $|\mathcal{N}| = 20$



(c) $|\mathcal{N}| = 25$

Source: Elaborated by the author.

3.4 Summary

This chapter investigated RQ1: *Are traditional metrics of fitness landscape analysis appropriate to characterize AutoML problems?* We have investigated the use of four metrics, namely Fitness Distance Correlation, Dispersion Metric, Fitness Entropic Measure, and Neutrality rate, with various parameters configurations and three different representation and distance metrics for the search space.

We observed that traditional metrics have several limitations for FLA in AutoML problems, and this is mainly due to the nature of the space, which is multimodal and

has several types of hyperparameters (continuous, discrete and conditional), making the definition of distance a challenge. Note that our search space is combinatorial, as all continuous attributes were discretized, but the hyperparameters still present dependencies among themselves, i.e., the choice of one parameter is conditioned by others.

FDC depends on two decisions: (i) the distance to the nearest global optima or the nearest local optima and (ii) the definition of distance. In all cases, the results obtained by changing these components indicate that the metric results are statistically different from each other in different scenarios, making it difficult to reach any conclusion and questioning the robustness of this metric.

FEM obtained very different results from the ones presented in its original paper [17], with extremely high entropy. This is because the space is combinatorial, while the metric was originally designed for continuous landscapes. Although adapting this metric for a combinatorial space requires only changing of the way the random walk is generated, more analyses are needed to understand the behavior of the metric in combinatorial spaces.

DM was also affected by the concept of distance and each method used presented different results. While the *ad hoc* and euclidean distances indicate that DM varies from positive to negative, the hamming distance shows DM as completely negative. The magnitude of the values is also different and the impact of distance is significant.

Due to these characteristics, we conclude that traditional metrics are not suitable to characterize AutoML search spaces, as most metrics were developed to either continuous or combinatorial spaces that do not have any hierarchical structure in their definition.

Chapter 4

Local Optima Network for Analyzing AutoML Search Spaces

In the previous chapter we have showed that standard metrics for FLA are not appropriate to analyze the search space of AutoML problems. Regardless of being analyzing AutoML problems, FLA usually present some common limitations. First, they focus mainly on extracting local measurements from the fitness landscape, which can be restrictive and miss the global view of the space [32, 34]. Second, most metrics are computationally expensive and, per definition, do not focus their analysis on the most relevant regions of the search space, such as the ones close to the local optima [24]. In this direction, this chapter addresses RQ2: *Compared to the traditional metrics of FLA, are Local Optima Networks (LONs) more appropriate to characterize AutoML spaces?*

For that, we show how to use LONs and their variants – MLON and CMLON – to analyze the search space of AutoML problems. As the original version of LON is known for not supporting search spaces with neutrality – and the literature has previously discussed that AutoML search spaces can have high neutrality [34, 32] – MLONs and CMLONs are used to assess the neutrality of the space.

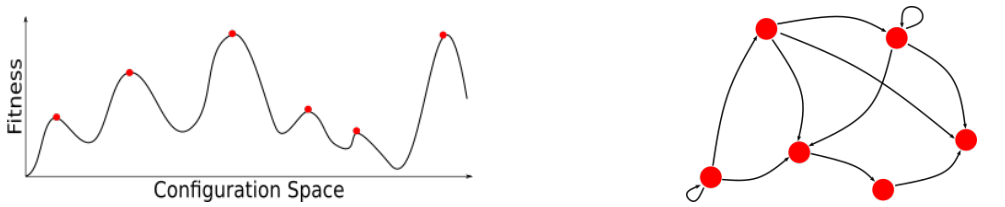
4.1 Methodology

Starting from the same definition of fitness landscape presented in Section 3.1, here we explain how we built the LONs for analyzing the AutoML fitness landscapes.

Recall that a LON is a graph built from the fitness landscape of a problem, ie, $LON = (V', E')$, where V' corresponds to the local optima and $V' = \{v \,|\, v \in V \text{ and } \mathcal{F}(v) \geq \mathcal{F}(\mathcal{N}(u))\}$ and $\mathcal{F} : S \rightarrow \mathbb{R}$ is a function that maps each solution in the search space to a quality metric, in our case, the fitness. Figure 4.1 illustrates the concept: the local optima of the fitness landscape are represented as vertices of the LON. Building the LON highly depends on the concept of neighborhood, $\mathcal{N}(u)$), as discussed

next.

Figure 4.1: Illustration of LON resulting from a fitness landscape



(a) Fitness Landscape (b) Local Optima Network

Source: Elaborated by the author.

4.1.1 Neighborhood Operator and Mutation Operator

In this chapter, we chose to use the original representation of the tree to represent each solution of the search space. From there, we defined the neighborhood using the simplest possible method: by using the concept of mutation, in the same way as done in [32].

Figure 4.2 shows an example of mutation, where the classification algorithm was changed. Mutation occurs through the random selection of a node in the tree, where the probability of a node being selected is inversely proportional to its distance from the root of the tree, and generates a subtree from that node.

After some experiments, we observed that the cost of mutating a pipeline was a bottleneck to the experiments. As an alternative, we used the strategy of calculating the Probability Mass Function (PMF) of the mutation of pipeline u to generate v and generate the mutations according to the PMF. The probability of mutating a node is proportional to the product of selecting a given node by the number of subtrees that can be built from it. This process can be repeated until reaching the root of the tree, where the probability of generating the pipeline v is the probability of generating any tree, that is, $p = 1/\#\text{tree}$.

The process is illustrated in Algorithm 1, which calculates the probability of mutation between a pair of pipelines. Initially the algorithm looks for nodes with different values in the trees passed as an argument and returns them. Then, the algorithm identifies the path from the node to the root common to all nodes identified by the previous function. In the figure, these nodes are represented by the shaded background color. Next, for each of these nodes present in the path, the function accumulates the probability considering the probability of the node being selected (`GET_PROB_SEL`) multiplied by

the probability of a subtree of v rooted at the node common to both pipelines is generated (`GET_NUM_COMB`).

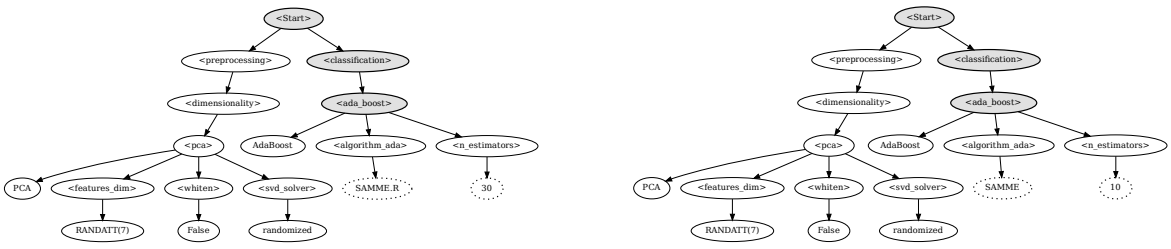
Algorithm 1 Mutation probability

```
1:  function MUTATION_PROB(p1, p2)
2:      diffNodes ← FIND_DIFF_NODES(p1, p2)
3:      mutationPath ← COMMON_ANCESTOR(diffNodes)
4:      prob ← 0
5:      for node in mutationPath do
6:          probSel ← GET_PROB_SEL(node)
7:          numComb ← GET_NUM_COMB(node)
8:          prob += probSel · 1/numComb
9:      end for
10:     return prob
11: end function
```

The neighborhood $\mathcal{N}$ of a pipeline p is then defined as all pipelines generated from p when applying the mutation operator. A consequence of the neighborhood definition is that neighboring pipelines are more likely to have the same algorithms, since the nodes most likely to be selected are the leaf nodes, which represent the hyperparameters. Notice that this is a desirable property for this operator, since changing a hyperparameter of an algorithm usually has less impact than changing an entire algorithm.

However, it is still possible to pipelines with completely different algorithms be neighbors. For example, Figure 4.2 shows the process of calculating the probability of mutation between two pipelines. Observe that the point selected for mutation is the `<ada_boost>` and, from it, an entire subtree is generated according to the grammar. Note, however, that if any other parent node, direct or not, of `<ada_boost>` were selected (the nodes highlighted by the shaded background color), it would be possible to get the same neighbor if the same subtree was generated, but the probability of selecting nodes closer to the root is smaller than that of selecting nodes further away from the root.

Figure 4.2: Examples of neighbor pipelines.



(a) Pipeline 1 (b) Pipeline 2

Source: Elaborated by the author.

As we are working with an enumerable space, we evaluate all solutions and identify

the LO in the LON instead of sampling the space – which is the standard for larger spaces. In these cases, the sampling process is performed using an Iterated Local Search (ILS).

It is important to say that the definition of neighborhood affects the analysis of the fitness landscape as it describes how it is explored. In this work, the neighborhood has (i) fixed size and (ii) there is no repetition of neighbors, that is, each solution in the configuration space has n distinct neighbors. These definitions are unrealistic for most (if not all) AutoML algorithms. However, this work contributes to (i) the understanding of metrics in a constrained space and (ii) the development of a representation capable of covering a wide range of AutoML algorithms, whether using Genetic Programming (GP) or Bayesian optimization.

4.1.2 Basins of Attraction and LON Edges

Having the LO, we need to define the concept of basins of attraction. A basin of attraction is formed by all solutions in the space that are "attracted' to a given local optimum, that is, all solutions that, after a local search (LS), end in LO_u. Formally:

$$\text{Basin of attraction LO}_\text{u} := \{v \in V \mid LS(v) = LO_u\} \tag{4.1}$$

where $LS : S \to S$ performs a local search: given a solution, it returns its LO. In practice, LS is usually defined as the Hill-Climb (HC) [28, 25, 2]. The HC can be implemented in different ways, including the use of first improvement, best improvement, worst improvement, approximate worst improvement and max-expansion. The results in [43] show max-expansion as the most effective to find high quality solutions. Here we use the best improvement due to its simplicity and the use of other strategies is left for future work.

Given the definitions above, observe there is a direct relationship between the number of LO and the roughness of the search space. According to the literature, roughness is a characteristic that can make search difficult for different search methods [18], specially those that follow a local search approach. As the LON is a graph, we can then extract a lot of metrics from the network analysis literature to characterize the search space of the generalized CASH problem.

Understanding the relationship between basins of attraction, i.e., the ability or probability of a solution go from one basin of attraction to another, can help to understand how space is structured. The relationship between basins of attraction is represented by weighted directed edges between LO.

Given an edge (LO_u, LO_v) that connects two local optima LO_u and LO_v, *basin-transition edges* receive weights given by the sum of the probabilities of a solution in the

basin of attraction of LO_u end up in the basin of attraction of LO_v after a perturbation (mutation) is applied to the solution. That is,

$$p(LO_u, LO_v) = \frac{1}{|LO_u|} \sum_{u \in LO_u} \sum_{v \in LO_v} p(u \to v)$$

where $p(u \to v)$ is the probability of mutation of a solution u in the basin of attraction of LO_u end up in v, that is, in the basin of attraction of LO_v.

4.1.3 Monotonic LON (MLON) and Compressed MLON (CMLON)

Although the original definition of LON reduces the configuration space by looking only at the LOs, in many problems, even the space composed only of LOs, still has a high number of nodes and edges, making it difficult to visualize the problem and not providing means for analyzing the neutrality of the space. Because of that, the Monotonic LON (MLON) was proposed. The main difference of the MLON compared to the original LON is that it does not include edges where the fitness of the source node is less than of the target node, i.e., a LO only connects to another with improved fitness values. Hence, MLONs are sparser than the original LON.

As in many cases the reduction from LON to MLON may not be enough to reduce the space complexity and there may still existing edges between nodes with the same fitness, Compressed MLONs (CMLON) were introduced. CMLONs compress adjacent nodes that have the same value of fitness in the MLON. This type of compression is very effective in networks with high neutrality, i.e., with low variations of fitness. Neutral regions of the network are identified using an adaption of a Breadth First Search (BFS), which receives a node u and returns the nodes with the same fitness and that can be reached from u.

The regions where LOs are compressed for having the same fitness are known as plateaus [25, 2], and the sets of LOs that are in the basins of attraction of other LOs after edges with decreasing fitness are removed are named funnels [25].

4.1.4 Metrics for analyzing LONs

Having defined the LONs that represent the search spaces of interest, we can use of set of metrics to analyze the structure of a LON, MLON, and CMLON [25]. We used a set of 10 metrics for that (5 from LON and 5 from CMLON).

Additionally, for each type of edge being considered in the standard LON representation, the following metrics are analyzed:

- **noptima**: is the number of local optima, and directly related with space roughness;
- **nglobal**: is the number of global optima, and indicates the space modality;
- **edgesi**: measures the sum of the weights of the edges that leave a solution u to a solution v, where $\mathcal{F}(u) < \mathcal{F}(v)$. Together with other edge metrics, it helps to understand the space dynamics, ie, how the optimization flows;
- **edgesn**: measures the total weight of neural edges, i.e., edges between nodes with same fitness; and
- **edgesw**: measures the total weight of edges that connect solutions in nodes u and v, where $\mathcal{F}(u) > \mathcal{F}(v)$.

Apart from the metrics related to traditional LONs, five other can be extracted from CMLONs:

- **ncoptima**: is the number of compressed LO, i.e., the number of LO that are neighbors of other LO with the same value of fitness and were merged in the CMLON;
- **ncglobal**: a subset of *ncoptima* that accounts only for the number of global optima compressed;
- **ncedges**: calculates the weight of edges between the local optima that were compressed when building the CMLON from the MLON.
- **neutrality**: given by the ratio between the number of compressed LO and the number of LO in the space;
- **lplateau**: measures the size of the plateaus – compressed regions of the CMLON – and identifies the regions with the highest number of LO.

4.2 Results and Discussion

Again, recall that the experiments considered the same 20 fitness landscapes introduced in Section 3.2. All experiments were repeated 30 times for statistical purposes. Three values of neighborhood size are reported here: 15, 20 and 25. Notice that other values of neighborhood, including 15, 20, 25, 30, 50 and 100, were also tested but omitted here since the larger the neighborhood, the lower the performance of the optimization (which generally seeks to reduce the number of solutions evaluated). These results were reported in our previous work [45].

Before looking at the metrics, we first verified if the size of the basins of attraction from different executions come from the same distribution. It is important to check whether small changes in the input graph cause large changes in the LON. For that, a two-sample Kolmogorov-Smirnov test was applied to each LON built for each of the 3 neighborhood sizes. The results show that 86.31% of the 120 experiments have p-value smaller than 0.001. This indicates that the structure of the LON undergoes variation in different samplings of the fitness landscape evaluated in a given dataset.

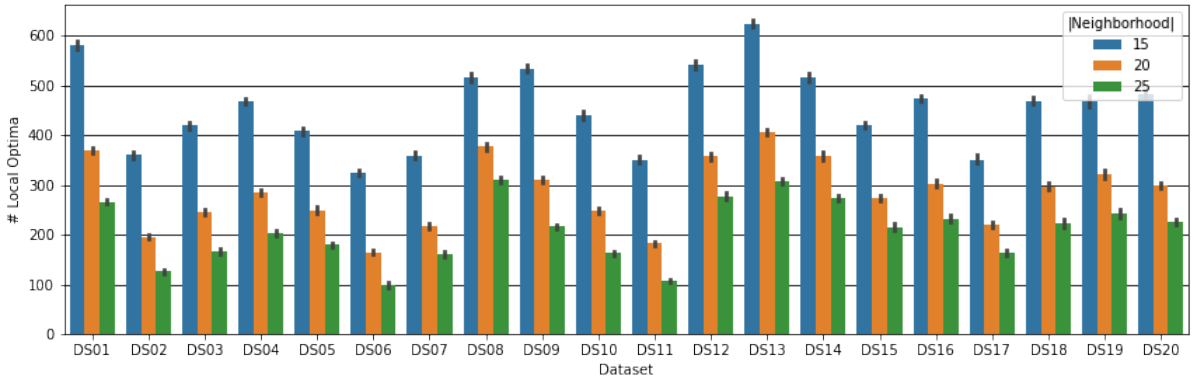
This section is divided into two parts according to the characteristic of the fitness landscape being analyzed. Roughness is addressed in Section 4.2.1, where the number of local optima and the size of basins of attraction are analyzed. Neutrality is addressed in Section 4.2.2, where the metrics extracted from the CMLON are discussed.

4.2.1 Roughness Analysis with LONs

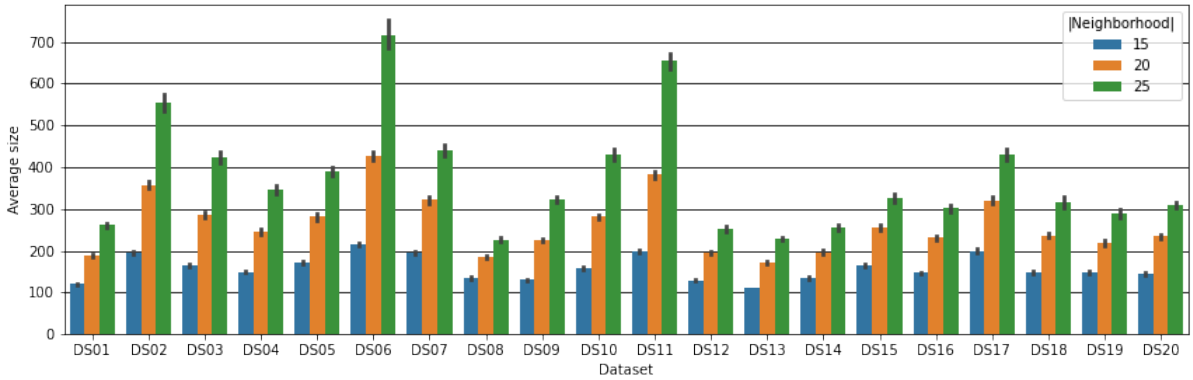
Turning to the analysis of the LON obtained from the 20 datasets studied in this paper, Figure 4.3a shows the number of LO per dataset. Observe that by changing the neighborhood size we reduce the number of LOs in all datasets. However, this reduction is more significant in a few datasets. For example, for dataset DS06, the reduction in the number of LO is of 69.50%, while for dataset DS08 we get a reduction of 40.06% when changing the neighborhood size from 15 to 25. This shows that increasing the neighborhood size can bring significant gains for some datasets. Also note that the reduction in the number of LOs when neighborhood size increases from 15 to 20 is higher than when it goes from 20 to 25, suggesting there might be an ideal trade-off between neighborhood size and space roughness. Recall that space roughness is related to the number of peaks, valleys and other anomalies in the fitness landscape and this can be measured through the number of nodes in the LON.

Figure 4.3b shows the size of the basins of attraction, and is complementary to Figure 4.3a. Since the number of AutoML solution configurations is constant and equals to 69,960, as the number of LO decreases the concentration of solutions in each basin of attraction increases. Increasing the neighborhood size causes many LOs to be attached to others, increasing the number of solutions in a basin and creating "super" basins of attraction.

Figure 4.3: LON features for the 20 datasets.



(a) Number of local optima.



(b) Size of basin of attraction.

Source: Elaborated by the author.

4.2.2 Neutrality Analysis with MLON and CMLON

As previously explained, MLONs and CMLONs have the advantage of measuring the neutrality of the space. The results of the metrics extracted from these networks are discussed in this section. We show the metrics extracted from the neighborhood of size 15 because, as discussed later in this section, the correlation between the results of

different neighborhood sizes is high and the analyses of the observed patterns similar (or equal) to those presented for this specific size. Table 4.1 indicates that the DS13 dataset has the highest number of compressed local optima: 544.60 on average. As compression only occurs between adjacent solutions (which have an edge with weight greater than zero in the LON domain), this indicates that there is a region in the configuration space where, when escaping from a given basin of attraction, it can lead to another basin of attraction with the same fitness. This scenario means that the optimization algorithm can get "stuck" in the search for an optimal solution always reaching the same final fitness.

In the special case of compressed global optima (ncglobal) it is possible to notice that DS06 has the largest number, although some datasets have only one global optima and, this metric is zero in this case. The presence of space-concentrated global optima can be both positive and negative. An advantage is that after finding one solution, finding alternative solutions can be easier, as basins of attraction are more likely to have a fitness as good as the current one, and the evaluation cost of the ML pipeline found can be much more efficient for the user than the current. One drawback is that if all optimal solutions are concentrated into a specific region of space, finding any global optimum is as difficult as finding the optimum of a unimodal space.

The simple fact that they are compressed may not indicate the intensity of "attraction" that these local or global optima have for each other. There may be cases of a not very intense attraction and the advantages/disadvantages identified above are not valid in these spaces. Therefore, analyzing the total weight of the edges that connect the optima compressed is a way of measuring the degree of attraction between them.

In this case, DS20 is the one with the highest total weight of compressed edges, equals to 0.74, which is 4.6 times larger than the values of weight of the second dataset with the most compressed edges, DS08. In DS20, there is a high chance that a search algorithm will escape from a basin of attraction and reach another one formed by an optimal solution with the same fitness as the current one. This characteristic can make the space exploration difficult, since algorithms that use local search can get stuck in these basins without finding anything new.

Regarding neutrality rate, DS17 has the highest value, 0.20, although there are other datasets with a similar neutrality rate. This metric indicates the proportion of local optima compressed in space, therefore, 1/4 of the local optima in DS17 have some degree of neutrality – it is not possible, however, to define the intensity of neutrality through the metric. Based on the results, 17/20 datasets have a neutrality rate greater than 0.1, that is, there is neutrality in one-tenth of the local optima present in the configuration space.

The *lplateau* metric shows the size of the plateaus, where it is possible to identify plateaus of up to 67.40 solutions, as is the case with dataset DS13.

As we increase the size of the neighborhood, we observe the same behaviour found in the aforementioned analyses. To confirm that, the Spearman correlation (ρ) between

the values of the metrics as neighborhood size grows, i.e., if X_{N_a} is the value of the metric X obtained in the graph with neighborhood size N_a, here the correlation is measured between $\rho(X_{N_a}, X_{N_b})$. Since there are 6 metrics (excluding edgesi, edgesw and edgesn, since they are not related with CMLON), 3 neighborhood sizes, and the Spearman correlation is commutative, this yields $6 \times \binom{3}{2} = 18$ different results. To simplify the analyses, correlations greater than 0.8 are considered as a high correlation, which happens in 15 cases and with p-value less than 0.00001. Hence, we can reject the null hypothesis – that there is no correlation – in all cases. This means that there is a similarity between the results obtained for different neighborhood sizes and, to avoid redundancy, our discussion is limited to neighborhood size 15.

Following the analysis of the metrics, some may be related to similar characteristics, such as the number of compressed local optima, the number of compressed edges and others. To verify if there is any relationship between these metrics, the Spearman correlation was calculated between all metrics for each graph constructed. The results are shown in Figure 4.4. In the figure, darker values indicate negative correlations and lighter values indicate positive correlation. The results show that for neighborhood size 15, most of the metrics have little correlation, except *nlocal* and *ncoptima* metrics: the number of local optima in the space has a strong correlation with the number of compressed local optima. However, as the size of the neighborhood increases, the correlation between the metrics (i) *lplateau* and *ncoptima* and (ii) *lplateau* and *nlocal* also increase. An increase in neighborhood size also impacts the negative correlation between *lplateau* and *neutrality*, indicating that as the neutrality rate decreases, the size of the largest plateau increases,

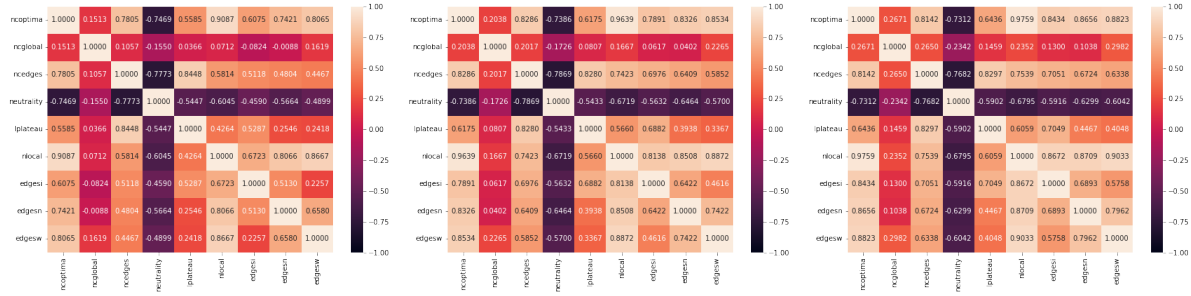
Table 4.1: CMLON of configuration space with neighborhood size 15.

Dataset	ncoptima	ncglobal	ncedges	neutrality	lplateau	nlocal	edgesi	edgesw	edgesn
DS01	458.57	0.33	28.46	0.13	33.50	581.30	252.61	253.47	65.16
DS02	230.93	4.23	13.18	0.17	17.80	360.47	221.45	86.93	46.52
DS03	302.50	4.80	25.47	0.14	52.07	420.13	238.11	111.16	61.07
DS04	388.77	8.80	28.77	0.11	35.33	468.43	261.83	140.40	59.04
DS05	308.73	12.40	20.00	0.14	25.50	408.67	171.63	173.44	57.06
DS06	193.20	5.13	15.82	0.16	16.33	325.60	195.17	65.00	52.47
DS07	280.53	0.47	24.55	0.13	26.27	359.50	211.83	91.93	50.74
DS08	474.03	24.00	59.54	0.07	79.47	517.10	246.08	200.81	65.48
DS09	356.03	8.67	22.95	0.14	51.67	534.37	227.34	231.29	57.10
DS10	301.43	5.10	15.58	0.16	24.37	440.57	244.06	125.58	65.95
DS11	158.63	0.00	5.79	0.15	11.67	350.93	216.90	74.04	52.36
DS12	463.63	5.00	40.20	0.10	46.53	542.13	301.10	165.29	69.55
DS13	543.03	2.90	57.65	0.07	87.17	624.47	281.18	242.98	91.84
DS14	468.17	0.00	67.65	0.07	83.93	516.63	240.92	200.50	70.92
DS15	353.80	11.90	31.62	0.11	46.90	421.97	230.81	136.71	49.01
DS16	381.27	4.23	29.91	0.13	32.07	474.27	217.55	193.23	57.62
DS17	292.77	4.00	24.76	0.15	25.83	351.20	152.54	144.29	49.80
DS18	373.83	0.10	32.41	0.12	37.70	469.33	227.24	172.54	63.69
DS19	409.47	1.67	68.76	0.12	176.40	469.20	287.67	127.28	52.59
DS20	396.47	9.87	33.49	0.12	38.97	482.93	197.37	211.52	68.00

Source: Elaborated by the author.

suggesting the formation of a large plateau.

Figure 4.4: Spearman correlation (ρ) between the metrics present in Table 4.1. The figure on the left corresponds to the graph with a neighborhood size 15, the center with neighborhood size 20 and, on the right, neighborhood size 25.



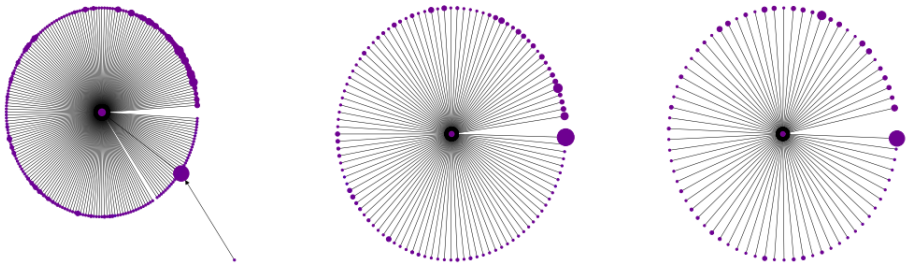
Source: Elaborated by the author.

Figures 4.5 and 4.6 show a representation of the CMLON structure of two datasets: DS09 (ml-prove) and DS15 (statlog-segment), respectively. In these figures, the vertex radius is proportional to the number of compressed local optima. The color of the vertices is proportional to the fitness of the local optimum that formed the basin of attraction, where purple/red indicates that the fitness is +1 and black indicates that the fitness is 0. As the number of edges is very large, it is difficult to visualize the graph. Hence, only the edges that make up the shortest path between the compressed local optimum and the global optimum are kept. As can be seen, the graphs are fully connected, although some funnels are not directly connected at the global optimum when the neighborhood is small. In all cases, however, increasing the neighborhood creates a connection between these funnels and the global optimum.

In the CMLON of DS09, observe that the funnel of the global optimum is relatively small and that there are local optima with funnels larger than the one of the global optimum. In this way, the search algorithm can get stuck in a suboptimal region of space and this translates into space difficulty. In the graphs in the figure, only 3.95% of the edges were illustrated in the first graph, 5.11% on the graph in the center and 5.22% in the graph on the right. In Figures 4.5 and 4.6, the vertices are plateaus in the CMLON, i.e., compressed LO. Edges only indicate the direction of the path. Note that during the CMLON generation process only the improving edges are kept, so the central vertex (sink in graph terminology) in the figures represents the global optimum.

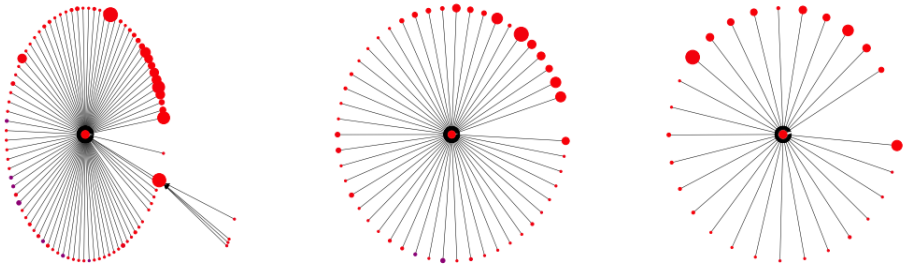
Figure 4.6 shows the funnels of DS15. In this dataset, in contrast to DS09, there are several funnels with relatively similar sizes. Another characteristic observed in both datasets is that increasing neighborhood makes all funnels have a direct connection with the global optimum. In this figure, the first graph has 8.44% of the edges present in the CMLON and the others have 8.54% and 10.63%, respectively.

Figure 4.5: CMLON representation of dataset ml-prove (DS09). Only the edges in the shortest path to the global optimum are represented.



Source: Elaborated by the author.

Figure 4.6: CMLON representation of dataset statlog-segment (DS15). Only the edges in the shortest path to the global optimum are represented.



Source: Elaborated by the author.

4.3 Summary

This chapter investigated RQ2: *Compared to the traditional metrics of FLA, are Local Optima Networks (LONs) more appropriate to characterize AutoML spaces in terms of space roughness and neutrality?*

Using the concept of LONs, we model the fitness landscape of AutoML problems and extracted metrics of difficulty, such as roughness, obtained by analyzing the number of local optima present in space. We have also used a variation of the traditional LON, namely CMLON, to look at the space neutrality.

After analyzing a set of 6 metrics for LON and 5 metrics for CMLON, including the total weight of compressed edges and the number of compressed global optima, we concluded that the space difficulty can vary substantially according to the dataset used, as neutrality and roughness can double in value from one dataset to another. This means

that an algorithm that has performed well in optimizing in one dataset may present poor performance in another, even when dealing with the same problem. One never escapes the no-free-lunch [10].

However, when compared to the standard FLA metrics discussed in Chapter 3, LONs seem more appropriate to assess the characteristics and measure the difficulty of AutoML search spaces because, apart from given a global view of the space, empirical results are more robust.

Chapter 5

Conclusions and Future Work

This work investigated two different approaches to characterize the fitness landscape of AutoML problems, more specifically for the case of automatically evolving machine learning pipelines. One of the main challenge was to define the fitness landscape itself, once deciding the concept of neighborhood and distance in these problems is not straightforward. But having the fitness landscape, we focused on the two research questions posed in Chapter 1:

1. Are traditional metrics of fitness landscape analysis appropriate to characterize the search space of AutoML problems?

2. Compared to the traditional metrics of FLA, are Local Optima Networks (LONs) more suitable to characterize AutoML spaces?

In order to answer the first question, we followed the standard way FLA is performed in most problems, where metrics are calculate to estimate the roughness, modality, and neutrality of these landscapes. Considering standard metrics of FLA, four metrics were tested, namely FDC, DM, FEM, and neutrality rate. FDC is a classic metric for landscape analysis, and several experiments were carried out considering both the distance to the local optimum and to the global optimum as a reference. We observed that, depending on how the metrics that define the search space are defined, FDC generates completely different results. DM presented a behavior similar to that of FDC, with its results heavily impacted by the adopted configuration. FEM presented a high value when compared to the paper that introduced the metric mainly due to the fact that the metric was designed for continuous landscapes and was not able to generate useful results for non-continuous landscapes. The neutrality rate showed that certain datasets (DS16, DS13, DS05) have significantly higher neutrality than others, but this result depends on the δ tolerance, which, in turn, is affected by the fitness variance of the space.

In conclusion, our answer to RQ1 is: Given the current definitions of AutoML search spaces, traditional metrics did not seem appropriate to characterize the problem. Perhaps once we have a good definition of neighborhood in spaces with conditional parameters these metrics can be adapted to perform better analysis.

To answer RQ2, we used the concepts of Local Optima Networks to perform FLA. Regarding the experiments with LONs, several metrics coming from the domain of complex networks were tested. Characteristics such as the number of local optima and the size of the basins of attraction help to analyze the roughness of the space, while the metrics extracted from the CMLON help to analyze the neutrality of the space. Our first experiment showed that the LON structure is unstable, that is, the distribution of local optima is impacted by sampling the fitness landscape when evaluated on the same dataset. It also showed that increasing the size of the neighborhood does not generate a linear smoothing in the number of optima in the space, so the cost of increasing the size of the neighborhood should outweigh the gain of reducing the roughness of the space. Space neutrality does not seem to be influenced by the size of the neighborhood, as the results found through the CMLON indicate that the relationships were maintained through the high correlation between them. The correlation between the CMLON metrics shows that there is a certain independence between them. Finally, the visualizations show that the number of (strictly) compressed local optima is greater than that of global ones, highlighting the probability that many search algorithms get "stuck" in their basin of attraction.

In summary, our answer to RQ2 is: LONs seem more appropriate to characterize these spaces, although the variation in the distribution of local optima due to the sampling carried out should be explored in future works. Several analyzed spaces are multimodal and others show accentuated neutrality when compared to others. The CMLON results show that there are regions that are not directly neutral, but that would be difficult for an ILS strategy, for example, to perform a good optimization locally.

The answers obtained in this work are only the first step towards understanding AutoML search spaces with greater depth. Being an exploratory study, there are many other directions that can be followed, as detailed in the next sections.

5.1 Study of Neighborhoods in Conditional Hyperparameter Spaces

It is clear at this point that one of the critical aspects of AutoML problems is defining the fitness landscape itself. The notion of neighborhood is not clear, and *ad hoc* methods such as the one proposed in [32] and also explored here is far from the ideal.

Conditional hyperparameter spaces are not a exclusivity of AutoML. In general, CASH problems also deal with it. Further studies on how to better represent neighborhoods on conditional spaces are essential to advance further on this matter.

5.2 Analyzes of Other FLA Metrics

The four metrics of exploratory landscape analysis used were selected based on our knowledge of the problem and how often they were explored in the literature. However, this does not mean other alternatives should not be considered. A more in-depth study of other metrics is a natural follow up of this work.

5.3 Assessing Problem Difficulty

There is no doubt that one of the most interesting points of analyzing fitness landscapes is to design search methods that can take better advantage of them. However, there is another line of work which may be extremely interesting in the context of AutoML: assessing problem difficulty.

There is a well-established area in machine learning, namely meta-learning [57], which has as its main objective to characterize the difficulty of machine learning problems. In meta-learning, learning problems are characterized in order to select the best algorithm to solve the problem at hand.

AutoML has the same objectives as meta-learning, in the sense that the final goal is to select the best machine learning algorithm/pipeline to handle a problem. When we start characterizing the fitness landscapes of AutoML problems, the underlying learning problem does have an impact on the difficulty of the landscape. Hence, looking at meta-learning features to characterize machine learning problems and understanding how to relate them to the landscape can help exploring the other ways to define problem difficulty.

5.4 Traditional and New AutoML Algorithms to Explore AutoML Spaces

Finally, the main objective of FLA is to allow new algorithms to be design and account for the characteristics of the space being explored. So far, we have not looked at how the algorithms explore this space. This is interesting as it may propose small

changes to current methods or even create new hybrid methods that can better explore the characteristics of these search spaces.

Bibliography

[1] Geir Agnarsson and Raymond Greenlaw. *Graph theory: Modeling, applications, and algorithms*. Prentice-Hall, Inc., 2006.

[2] Francisco Chicano, Gabriela Ochoa, and Marco Tomassini. Real-like MAX-SAT instances and the landscape structure across the phase transition. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 207–215, New York, NY, USA, jun 2021. ACM.

[3] Christopher W Cleghorn and Gabriela Ochoa. Understanding parameter spaces using local optima networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1657–1664, New York, NY, USA, jul 2021. ACM.

[4] Zbigniew J. Czech. Statistical measures of a fitness landscape for the vehicle routing problem. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE, 2008.

[5] Alex GC de Sá, Walter José GS Pinto, Luiz Otavio VB Oliveira, and Gisele L Pappa. Recipe: a grammar-based framework for automatically evolving classification pipelines. In *European Conference on Genetic Programming*, pages 246–261. Springer, 2017.

[6] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[7] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

[8] Unai Garciarena, Roberto Santana, and Alexander Mendiburu. Analysis of the Complexity of the Automatic Pipeline Generation Problem. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, jul 2018.

[9] G.Shala, A. Biedenkapp, N.Awad, S. Adriaensen, M.Lindauer, and F. Hutter. Learning step-size adaptation in cma-es. In *Proceedings of the Sixteenth International Conference on Parallel Problem Solving from Nature (PPSN'20)*, September 2020.

[10] Yu-Chi Ho and David L Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570, 2002.

[11] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[12] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[13] Terry Jones, Stephanie Forrest, et al. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *ICGA*, volume 95, pages 184–192, 1995.

[14] Mark Lipsitch. Adaptation on Rugged Landscapes generated by Iterated Local Interactions of Neighboring Genes. In *4th International Conference on Genetic Algorithms*, 1991.

[15] Monte Lunacek and Darrell Whitley. The dispersion metric and the CMA evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*, New York, New York, USA, 2006.

[16] Katherine Malan and Andries Petrus Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

[17] Katherine M. Malan and Andries P. Engelbrecht. Quantifying ruggedness of continuous landscapes using entropy. In *2009 IEEE Congress on Evolutionary Computation*, pages 1440–1447. IEEE, may 2009.

[18] Katherine M. Malan and Andries P. Engelbrecht. Fitness Landscape Analysis for Metaheuristic Performance Prediction. pages 103–132. 2014.

[19] Bernard Manderick, Mark K. de Weger, and Piet Spiessens. The genetic algorithm and the structure of the fitness landscape. In *ICGA*, 1991.

[20] Peter Merz. Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms. *Evolutionary Computation*, 12:303–325, 2004.

[21] F.P. Miller, A.F. Vandome, and M.B. John. *Kendall Tau Rank Correlation Coefficient*. VDM Publishing, 2010.

[22] Daniel Molina, Manuel Lozano, Carlos Garcia-Martinez, and Francisco Herrera. Memetic Algorithms for Continuous Optimisation Based on Local Search Chains. *Evolutionary Computation*, 2010.

[23] Mario A Muñoz, Yuan Sun, Michael Kirley, and Saman K Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

[24] Matheus Nunes, Paulo M. Fraga, and Gisele L. Pappa. Fitness landscape analysis of graph neural network architecture search spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 876–884, New York, NY, USA, jun 2021. ACM.

[25] Gabriela Ochoa and Francisco Chicano. Local optima network analysis for MAX-SAT. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1430–1437, New York, NY, USA, jul 2019. ACM.

[26] Gabriela Ochoa and Nadarajen Veerapen. Neural architecture search: A visual analysis. In *International Conference on Parallel Problem Solving from Nature*, pages 603–615. Springer, 2022.

[27] Gabriela Ochoa, Nadarajen Veerapen, Fabio Daolio, and Marco Tomassini. Understanding Phase Transitions with Local Optima Networks: Number Partitioning as a Case Study. pages 233–248. 2017.

[28] Gabriela Ochoa, Sébastien Verel, Fabio Daolio, and Marco Tomassini. Local Optima Networks: A New Model of Combinatorial Fitness Landscapes. In *Recent advances in the theory and application of fitness landscapes*, pages 233–262. Springer, 2014.

[29] Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pages 66–74, 2016.

[30] Alan Owen and Inman Harvey. Adapting particle swarm optimisation for fitness landscapes with neutrality. In *2007 IEEE Swarm Intelligence Symposium*, pages 258–265. IEEE, 2007.

[31] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[32] Cristiano G. Pimenta, Alex G. C. de Sá, Gabriela Ochoa, and Gisele L. Pappa. Fitness Landscape Analysis of Automated Machine Learning Search Spaces. In *European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar)*, pages 114–130. 2020.

[33] Yasha Pushak and Holger Hoos. Algorithm configuration landscapes. In *International Conference on Parallel Problem Solving from Nature*, pages 271–283. Springer, 2018.

[34] Yasha Pushak and Holger H. Hoos. AutoML Loss Landscapes. *ACM Transactions on Evolutionary Learning and Optimization (TELO)*, 2022.

[35] Anna Rakitianskaia, Eduan Bekker, Katherine M Malan, and Andries Engelbrecht. Analysis of error landscapes in multi-layered neural networks for classification. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 5270–5277. IEEE, 2016.

[36] Anna Rakitianskaia, Eduan Bekker, Katherine M. Malan, and Andries Engelbrecht. Analysis of error landscapes in multi-layered neural networks for classification. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016.

[37] Christian M Reidys and Peter F Stadler. Neutrality in fitness landscapes. *Applied Mathematics and Computation*, 117(2-3):321–350, 2001.

[38] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.*, 2021.

[39] Hendrik Richter. *Fitness Landscapes: From Evolutionary Biology to Evolutionary Computation*, pages 3–31. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[40] Nuno M. Rodrigues, Katherine M. Malan, Gabriela Ochoa, Leonardo Vanneschi, and Sara Silva. Fitness landscape analysis of convolutional neural network architectures for image classification. *Information Sciences*, 609:711–726, 2022.

[41] Nuno M Rodrigues, Sara Silva, and Leonardo Vanneschi. A study of fitness landscapes for neuroevolution. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[42] Peter F. Stadler and Wolfgang Schnabl. The landscape of the traveling salesman problem. *Physics Letters A*, 1992.

[43] Sara Tari and Gabriela Ochoa. Local search pivoting rules and the landscape global structure. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 278–286, New York, NY, USA, jun 2021. ACM.

[44] Matheus Cândido Teixeira and Gisele Lobo Pappa. Analysis of Neutrality of AutoML Search Spaces with Local Optima Networks. In *11th Brazilian Conference on Intelligent Systems*, 2022.

[45] Matheus Cândido Teixeira and Gisele Lobo Pappa. Understanding AutoML Search Spaces with Local Optima Networks. In *Genetic and Evolutionary Computation Conference*, 2022.

[46] Kalifou René Traoré, Andrés Camero, and Xiao Xiang Zhu. Fitness Landscape Footprint: A Framework to Compare Neural Architecture Search Problems. nov 2021.

[47] German Treimun-Costa, Elizabeth Montero, Gabriela Ochoa, and Nicolás Rojas-Morales. Modelling parameter configuration spaces with local optima networks. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 751–759, 2020.

[48] Willem Abraham van Aardt, Anna Sergeevna Bosman, and Katherine Mary Malan. Characterising neutrality in neural network error landscapes. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1374–1381, 2017.

[49] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2625, 2008.

[50] Leonardo Vanneschi, Yuri Pirola, and Philippe Collard. A quantitative study of neutrality in gp boolean landscapes. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 895–902, 2006.

[51] Leonardo Vanneschi, Yuri Pirola, and Philippe Collard. A quantitative study of neutrality in GP boolean landscapes. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 895–902, 2006.

[52] Leonardo Vanneschi, Yuri Pirola, Giancarlo Mauri, Marco Tomassini, Philippe Collard, and Sébastien Verel. A study of the neutrality of Boolean function landscapes in genetic programming. *Theoretical Computer Science*, 2012.

[53] Leonardo Vanneschi, Marco Tomassini, Philippe Collard, Sébastien Vérel, Yuri Pirola, and Giancarlo Mauri. A comprehensive view of fitness landscapes with neutrality and fitness clouds. In *European Conference on Genetic Programming*, pages 241–250. Springer, 2007.

[54] Vesselin K Vassilev, Terence C Fogarty, and Julian F Miller. Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application. In *Advances in evolutionary computing*, pages 3–44. Springer, 2003.

[55] Sebastien Verel, Philippe Collard, Marco Tomassini, and Leonardo Vanneschi. Neutral fitness landscape in the cellular automata majority problem. In *International Conference on Cellular Automata*, pages 258–267. Springer, 2006.

[56] Sébastien Vérel, Fabio Daolio, Gabriela Ochoa, and Marco Tomassini. Local Optima Networks with Escape Edges. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 49–60. Springer, 2012.

[57] Ricardo Vilalta, Christophe Giraud-Carrier, and Pavel Brazdil. *Meta-Learning - Concepts and Techniques*, pages 717–731. Springer US, Boston, MA, 2010.

[58] Ian H. Witten and Eibe Frank. Data mining - practical machine learning tools and techniques, second edition. In *The Morgan Kaufmann series in data management systems*, 2005.

[59] Sewall Wright et al. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. 1932.

[60] Feng Zou, Debao Chen, Hui Liu, Siyu Cao, Xuying Ji, and Yan Zhang. A survey of fitness landscape analysis for optimization. *Neurocomputing*, 503:129–139, 2022.

Appendix A

Grammar

```
<Start> ::= <preprocessing> <classification>

# Preprocessing
<preprocessing> ::= <no_preprocessing>
                  | <bounding>
                  | <dimensionality>

## No Preprocessing
<no_preprocessing> ::= NoPreprocessing

## Bounding
<bounding> ::= <standard_scaler>

<standard_scaler> ::= StandardScaler <with_std> <with_mean>

<with_std> ::= True
             | False

<with_mean> ::= True
              | False

## Dimensionality reduction and feature selection
<dimensionality> ::= <selectKBest>
                   | <pca>

<selectKBest> ::= SelectKBest <features_dim>

<pca> ::= PCA <features_dim> <whiten> <svd_solver>

<features_dim> ::= RANDATT(1,ATT-1)
<whiten> ::= True
           | False
```

```
33
34 <svd_solver> ::= full
35               | arpack
36               | randomized
37
38 # Classification
39 <classification> ::= <logistic_regression>
40                    | <mlp>
41                    | <knn>
42                    | <random_forest>
43                    | <ada_boost>
44
45 <logistic_regression> ::= LogisticRegression <penalty> <fit_intercept>
46                                               <max_iter> <warm_start>
47
48 <penalty> ::= l1
49            | l2
50
51 <fit_intercept> ::= True
52                  | False
53
54 <max_iter>  ::= 100
55              | 300
56              | 500
57
58 <warm_start> ::= True
59               | False
60
61 <mlp> ::= MLP <max_iter> <activation>
62
63 <max_iter>  ::= 100
64              | 300
65              | 500
66
67 <activation> ::= identity
68               | logistic
69               | tanh
70               | relu
71
72 <knn> ::= KNearestNeighbors <k> <weights> <k_algorithm_and_leaf_size>
73                              <d_metric_and_p>
```

```
74
75  <k> ::= 1
76      | 5
77      | 15
78      | 25
79      | 35
80
81  <weights> ::= uniform
82              | distance
83
84  <k_algorithm_and_leaf_size> ::= brute
85                                | kd_tree 20
86                                | kd_tree 40
87                                | kd_tree 60
88                                | kd_tree 80
89                                | kd_tree 100
90                                | ball_tree 20
91                                | ball_tree 40
92                                | ball_tree 60
93                                | ball_tree 80
94                                | ball_tree 100
95
96  <d_metric_and_p> ::= euclidean
97                     | manhattan
98                     | chebyshev
99                     | minkowski 1
100                    | minkowski 2
101                    | minkowski 3
102                    | minkowski 4
103                    | minkowski 5
104                    | minkowski 6
105                    | minkowski 7
106                    | minkowski 8
107                    | minkowski 9
108                    | minkowski 10
109
110 <random_forest> ::= RandomForest <criterion> <bootstrap_and_oob>
111                                   <class_weight_Trees> <n_estimators>
112                                   <warm_start> <max_features> <max_depth>
113
114 <criterion> ::= gini
```

```
115             | entropy
116
117 <bootstrap_and_oob> ::= True True
118                       | True False
119                       | False False
120
121 <class_weight_Trees> ::= balanced
122                        | balanced_subsample
123                        | None
124
125 <n_estimators> ::= 10
126                 | 30
127                 | 50
128
129 <warm_start> ::= True
130              | False
131
132 <max_features> ::= sqrt
133                 | log2
134
135 <max_depth> ::= 10
136             | 30
137             | 50
138
139 <ada_boost> ::= AdaBoost <algorithm_ada> <n_estimators>
140
141 <algorithm_ada> ::= SAMME.R
142                   | SAMME
143
144 <n_estimators> ::= 10
145                 | 30
146                 | 50
```