Interactive Feature Extraction using Implicit Knowledge Elicitation : Application to Power System Expertise

Laure Crochepierre
Université de Lorraine, CNRS,
LORIA, F-57000 Metz, France
and Rte R&D, Paris, France
laure.crochepierre@rte-france.com

Lydia Boudjeloud-Assala
Université de Lorraine, CNRS,
LORIA, F-57000 Metz, France
lydia.boudjeloud-assala@univ-lorraine.fr

Vincent Barbesant
Rte R&D, Paris, France
vincent.barbesant@rte-france.com

Abstract

Industrial systems such as power networks are continuously monitored by human experts who quickly identify potentially dangerous situations by their experience. As current energy trends increase the complexity of day-to-day grid operations, it becomes necessary to assist experts in their monitoring tasks. This paper proposes an interactive approach to create human-readable analytical expressions that describe physical phenomena by their most impacting quantities. We present an interactive platform that brings experts in the training loop to guide the expression search using their expertise. It uses an evolutionary approach based on Probabilistic Grammar Guided Genetic Programming with expertly created and updated grammars. Interactivity is multi-level: users can distill their knowledge both within and between evolutionary runs. We proposed two usage scenarios on a real-world dataset where the non-interactive algorithm either provides (case 1) or not (case 2) satisfactory solutions. We show improvements regarding the solution's precision (case 1) and complexity (case 2).

1. Introduction and motivations

Inferring a symbolic representation from a set of observations is one of the human brain's fascinating abilities. It makes it possible to represent data information in a condensed way and provides a better understanding of complex phenomena, especially in physical sciences. Today, the problems tackled in science are even more challenging and often require an advanced level of specialization. Due to this increasing complexity, finding a direct mathematical formulation to obtain data insights is not straightforward, even with the large amount of data generated in modern experiments. The use of interactive Human-in-The-Loop (HiTL) approaches could then leverage the power of the machine and human intelligence to create these condensed representations.

Like in scientific discovery, industrial processes control and supervision face similar issues regarding implicit knowledge formalization. As these processes are intrinsically complex systems, they are heavily monitored by sensor networks to prevent them from going beyond their operating limits [1]. From these measurements, many complex simulation tools have been set up to assist operators' workload carrying out this monitoring. However, in cases such as one of the power systems we are focusing on in this paper, human expertise is still required, and heavily-trained operators inevitably perform in-depth analyses of the situations measured by these sensors. Consequently, a large proportion of domain knowledge is implicitly held by humans. In addition to that, electric power systems operation is also facing many challenges today. From the increase in renewable resources connected to the grid to the change of users' consumption habits and the development of European interconnections, Transmission System Operators (TSOs) have noticed an increase in real-time operations' difficulty [2]. For example, electrical power lines conducting electricity from production to consumers are operated closer to their physical limit, and thus, operators have to go through the network measurements analysis faster to keep time to handle more critical situations. Consequently, TSO companies need to develop new technical solutions to tackle the operators' new peak activities. As done today, the information synthesis from the sensor's measurements is computer-assisted by some hand-crafted aggregation indicators and computationally massive simulations. However, these simulations are quite long to compute and cannot cover all possible future forecasts. Operators, in addition, historically created indicators using their expert knowledge, but as is, indicators are not exhaustive and can not confirm the safety of all situations. Thus, automatically finding analytical solutions from grid measurements would help operators to perform this

URI: https://hdl.handle.net/10125/79544
978-0-9981331-5-7

information synthesis in the future. Here again, HiTL approaches would allow condensing complex dependencies between huge amounts of variables into easily interpretable expressions. Domain experts could then complete and improve the proposed interpretation of the phenomenon in light of their knowledge.

The artificial intelligence research community has long been working on automatic methods to solve the problem of explicit formulas extractions, also known as *symbolic regression* (SR)[3]. More precisely, SR is a type of regression analysis that searches a space of mathematical expressions to find a closed-form equation that best fits a given set of observations. Nowadays, SR is still mainly performed using Genetic Programming (GP) [4], a type of Genetic Algorithms (GA) which outputs symbolic expressions. However, standard GP algorithms tend to be slow to converge because of the large search space they have to explore. To avoid this drawback and tackle the challenges mentioned above, we designed a solution based on Probabilistic Grammar-Guided Genetic Programming (PG3P), an extension of Grammar-Guided Genetic Programming (G3P) which uses a Probabilistic Context-Free Grammar [5]. Grammatical rules are weighted in order to constrain the search and improve the convergence time. As PG3P provides human-readable solutions, we propose an interactive platform where human experts can directly analyze intermediary results, propose expertly created solutions, and provide feedback if necessary. Grammar is also constructed and updated interactively, which allows the user to describe his expertise in a notation the algorithm can handle. Interactivity here allows the domain experts to guide the search towards what is regarded as the most relevant region of the search space, reducing the computational effort required to infer highly relevant solutions.

Our contributions are the following :

- We propose an interactive process based on PG3P, which brings the expert into the learning loop twice to update the grammar both inside the learning loop and in-between evolutionary runs.
- We propose a web-based platform to extract domain knowledge through various direct interactions between experts and symbolic expressions.
- We propose two interactive scenarios on a real-world dataset and show that our expert HiTL approach improves the solution and reduces the complexity compared with non-interactive runs.

The rest of this paper is organized as follows. First, Section 2 summarize related state-of-the-art works. Section 3 gives a global data description. In Sections 4 and 5, we describe the proposed method by detailing both the algorithm methodology and the user interface. Section 6 provides two interactive scenarios, and some results obtained using this framework, and finally, Section 7 offers concluding remarks and perspectives.

2. Related Works

2.1. Feature extraction

In real-world applications, data we want to extract knowledge from are often initially measured in a high dimensional space, which redundant sparse information tends to lie on low dimensional manifolds and as such can be synthesized into a representation with fewer dimensions. We refer to the task of finding this manifold as Dimensionality Reduction (DR). More formally, given a set of observations with a high dimensionality D, DR finds a new representation with d dimensions, such that $d << D$. DR techniques are generally divided into feature selection (FS) and feature extraction (FE).

While FS aims at selecting a *subset* of the most relevant fetures [6], FE focus on forming a new feature space from a *combination* of some initial features. FE methods can either be linear or non-linear (NL). Linear methods like PCA[7] are often seen as more interpretable, but they also lack expressivity, which in many cases prevents them from capturing the underlying data structure. On the contrary, when expressiveness is favored, NL methods can offer more detailed representation. They offer diverse purpose, such as extending linear FE to NL-FE[8], visualizing data [9] or encoding data as distributions [10].

NL methods also have some drawbacks: mainly the difficulty of interpreting them and the complexity of adding user knowledge. Thus, we will describe in more detail Grammar-Based methods in Sections 2.2 and 2.3, which propose an interesting alternative to performing interpretable NL-FE with prior knowledge.

2.2. Symbolic Regression

When creating one feature ($d = 1$), NL-FE with interpretable combinations can be done by Symbolic Regression (SR). SR is defined as the task of finding a symbolic equation that best matches a set of inputs. It was initially implemented using GP methods [11] and found a wide range of applications [12, 13]. Today in the Deep Learning community, new approaches propose, for example, to encode the equation in the neural network structure and activation functions [14, 15], to use Recurrent Neural Networks to predict a string equation [16] or use Deep Reinforcement Learning (RL)

as a search engine [17]. However, those methods are often data-hungry, and few of them manage to include prior knowledge through sophisticated constraints, such as those included by context-free grammars [18]. Regarding RL approaches, SR tasks use sparse delayed rewards, as the metric is only evaluated at the end of an episode [17]. It isn't an ideal RL setup and could lead to convergence issues with complex real-world data.

2.3. Domain-knowledge in Genetic Programming

As stated above, Genetic programming (GP) was, until recently, one of the most common methods to perform SR. Koza [4] initially proposed GP as an extension of Genetic Algorithms (GA) for computer program induction: while a GA has a population of fixed-length binary vectors, GP evolves a population of programs represented as a tree. Due to the large search space, vanilla-GP has a slow convergence. Thus, various extensions propose to reduce this space with constraints. These constraints can both improve convergence [19] and impose a prior knowledge of the domain in the learning, to promote more coherent solutions.

Early on, Koza [4] sensed that syntactic constraints were an interesting solution to restrict the search space. Also, in the early days of GP, Montana proposed a Strongly Typed Genetic Programming approach [20], where variables, function arguments, and return value are given a specific data type. It was a first solution to enforce knowledge about the programming language structure. Another approach [21] took into account the physical dimensions of each individual and evaluating its distance to a correct dimension in the fitness function. However, forcing dimensional consistency as a distance does not prevent the final individuals from having incorrect dimensions. Alternatively, another recent work proposes to use ontologies [22] to elicit prior knowledge by adding new features to the initial feature-set.

Domain knowledge can also take the form of grammatical rules written in a Backus-Naur form (BNF) [23], as in Grammar-Guided Genetic Programming (G3P). A BNF grammar is composed of :

- a set of terminal symbols (the input features)
- a set of non-terminal symbols (the operators to combine features e.g. $\times$, +, ...)
- a set of production rules in the form `symbol` ::= `rule1` | `rule2` |... (how to perform operations on terminals)

On the left side of a production rule, separated by ::=, is the `symbol` produced by the application of a rule. On the right side, with | separator, are represented the alternative rules which can replace the `symbol`.

Early on, G3P was used for equation discovery (or re-discovery) [24] and since grammar-based evolution has been heavily exploited in real-world applications because of its great representational capacity: from seismic underground prospection [25] to glucose prediction in diabetic patients [26], and feature construction in High-Energy Physics [27].

2.4. Interactivity

As defined in [28], interactive Machine Learning (iML) is an interaction paradigm in which a user or user group iteratively builds and refines a mathematical model to describe a concept through cycles of input and review. Model refinement takes the user's insights as input through objects with many different forms. The model is then built in an Informed Machine Learning paradigm [29], where knowledge is given to the algorithm in a different form than the learning data such as algebraic equations, logic rules, or human feedback. In iML, iterations are more rapid, focused, and incremental than in traditional machine learning. Interactivity here enables a better understanding of the algorithm's results as the user has access to more information, which eventually improves trust in the proposed result. However, to be performing, several common user characteristics have to be taken into account, according to Amershi et al. [30]. First, users are humans, not oracles, and performing repetitive tasks like telling what is right and wrong to the computer could be perceived as annoying. Users also want to demonstrate how the algorithm should behave, which could be advantageous in our application. In addition, end-users naturally want to provide more than labels and could make suggestions in a variety of manners (e.g., suggesting new features or adjusting their importance). Therefore, users should have as many interaction types as possible. These fundamental elements are only possible if the users have at hand a carefully designed interface. Thus, we have tried to keep these elements in mind when developing the interactive platform.

In the GA community, interactive evolution (IE) models are mainly Interactive Genetic Algorithms (IGAs) or Human-Based Genetic Algorithms (HBGAs)[31]. First, IGA allows the user to assign a fitness score to individuals in the population. In HBGA [31], in addition to evaluating the fitness value, the user also performs all operations from initialization, mutation, crossover, to selection. IGAs are heavily exploited today in artistic or industrial creation [32, 33], where there is a need to draw out

the perception and subjective evaluation of the user regarding potential solutions. As the considered outputs are often impossible to compare with classical fitness functions, the user gives here a judgment-based fitness score. Eventually, regarding the use of interactivity for SR, Kim et al. proposed [34], an interactive platform where users can either approve or reject expressions found by a Deep Symbolic Regression mechanism. However, they do not ensure dimensional consistency.

One major issue in IE is the user fatigue induced by the repeated queries made by the algorithm. Strategies have been proposed to reduce it, such as using a relative evaluation of individuals [35] or training a deep learning model to map the user's fitness predictions [32]. Another solution is also to perform automated evaluation and occasionally ask for human evaluation every $n^{th} - generation$ [36]. As it is convenient not to overload the expert, we'll favor this last approach.

3. Data description

We built our platform as a SR-based interactive process to explain various phenomena on power grids. Our approach doesn't aim at replacing power flow (PF) or state estimation (SE) methods but rather proposes an alternative solution to create regularly updated indicators used for hazard pre-screening when PF and SE are too long to compute. It would be of particular interest in the so-called "N-1" studies, which simulate the disconnection of all electrical lines independently on a fixed time step. Thus, in a large network, an evaluation of a reduced set of indicators would be a much faster risk assessment solution.

An electrical power grid can be represented as a graph connecting a set of nodes by electrical lines. We especially focus on explaining power flows on several intricate power lines using sensors measurements and power network simulations. Let us denote by y the target features (flows) and X the set of input features (measurements and simulation results). Even if the power grid is hard to analyze as a whole, it can be divided into smaller, almost electrically independent regions to study the characteristics of each geographical zones [37]. In this paper, we restrict the study perimeter to a hilly mountain region in the French Alps made of 69 nodes and 92 lines. Only 9 lines connect the zone with other parts of the network. In this area, 24 y target power lines (or variables) were selected for analysis because of their operational difficulty. 13 are already well analyzed without interactivity (with a Pearson correlation score above 0.85). Ten others are harder to process without interactivity (score in $[0.65, 0.8]$).

We use 828 input features divided into 644 different types of measurements and 184 simulated features. Inputs contain various types of measurements (physical data: active power P, reactive power Q, voltage V at line extremities) as well as time-varying topological information relative to lines and bus nodes connectivity. Collected measurements ranged from January 2014 to December 2018 and resulted in a dataset made of 365,165 timesteps. Underlying in this work, we want to find simplified relationships between input and output variables of simulations performed independently on each time step. Thus, two consecutive time steps are considered independent as the values of the target variable y for two consecutive timesteps result from two static simulations computed independently using measurements from two different time steps.

The dataset is eventually separated into train and test sets where the PG3P algorithm uses the train set, and the test set is displayed in the visualization interface.

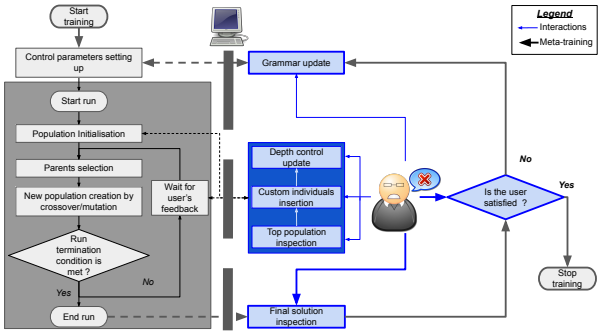

Figure 1. A high-level overview of the proposed platform with on the left, the GP evolution, and on the right, the multi-level interaction details. "start training" is the entry point of the process diagram.

4. System implementation

In Figure 1, we first give a high-level overview of the interactive platform. The training procedure is divided into two components: the PG3P algorithm based on IE on the left (detailed in Section 4.1), and the interactive knowledge elicitation on the right. In-between lies the interaction interface (see Section 5).

Our approach's specificity is the multi-level interactivity, which allows the user to distill his knowledge at various stages. The user interacts with the GP algorithm both during the run (through parameters update, individuals inspections, and insertion) and in-between two successive runs (mainly by updating the grammar). More precisely, during the evolutionary run, the user is queried for population initialization and population update. Successive runs are performed until the user is satisfied with the proposed solution.

```
<expr> ::= <p> | <s> || probs [0.5,0.5]
<p>    ::= <p>-<p> | <p>+<p> | abs(<p>) | <p_var>
           || probs [0.1, 0.3, 0.1, 0.5]
<q>    ::= <q>-<q> | <q>+<p> | abs(<q>) | <q_var>
           || probs [0.1, 0.3, 0.1, 0.5]
<p2>   ::= <p>*<p> | square(<p>) || probs [0.2, 0.8]
<q2>   ::= <q>*<q> | square(<q>) || probs[0.2, 0.8]
<s>    ::= sqrt(<p2> + <q2>) | <s>+<s> || probs [0.8, 0.2]
```

Figure 2. Probabilistic Grammar example with <p_var> and <q_var> as terminals.

4.1. Probabilistic Grammar

As detailed in the introduction, we are interested in interactively finding a non-linear mapping m between inputs observations X and a target variable y, which provides meaningful explanations about the relationships between variables to an expert user. Our approach uses PG3P as the core algorithm to learn this mapping. The main advantage we want to exploit from PG3P is the capacity of the grammar to hold information and structure about the problem because the resulting individuals will contain part of this structured knowledge. The restraining ability of the grammar is also very useful to speed up the search time.

We use Probabilistic Context-Free Grammars as an alternative to standard CFG (previously detailed in Section 2.3). In PG3P, a weight is associated with each rule of each production. The weight corresponds to the probability of the rule being sampled to replace its corresponding symbol. The probabilities are listed at the end of each line and separated from rules options by "||". For each production rule, the sum of probabilities over all alternative rules adds up to 1. This rule occurrence frequency restriction has the overall effect of reducing, even more, the search space and contributes to the reduction of the bloating phenomenon [38] where individuals' size explodes after a few iterations.

A simplistic version of a G3P grammar is provided in Figure 2, with only two terminals <p_var> <q_var>. The corresponding dimensions allowed to return are <p> or <s> with equal probabilities. The next two grammar lines describe how to obtain an individual of type <p> or <q>. The terminals are inserted there. Here a higher weight has been given to terminals so as to reduce recursivity and obtain shallower individuals.

The physical properties of the problem can also be described in grammar. First, we can define the physical units that are used in the algorithm. For example in the grammar from Figure 2, we define active power with symbol <p>, reactive power with symbol <q> and apparent power with symbol <s>. The grammar also details legal operations that can be carried out on each of the physical units. It is therefore not possible to add a power with a reactive power, but only two identical units : <p> + <p> or <q> + <q>. We can eventually add rules to explain how to go from one unit to another. For example, still in Figure 2, symbol <p2> (resp. <q2>) represents the square of an active (resp. reactive) power.

4.2. Individual representation

In PG3P and G3P, in general, an individual or program can be visualized as a tree where nodes are operations and leaves are terminals. To make the implementation more efficient, we use the representation of Canonical Grammatical Evolution, which encodes an individual as a list of integers representing a linear genome (also called chromosomes) [39]. Each integer in the genotype list matches the rule id in the current production. The phenotype (syntax tree) is then built from it by depth-first traversal, each rule id assigned to a tree node or leaf.

4.3. Fitness function

We want individuals to match as closely as possible the *behavior* of a target y. To handle all these requirements, we propose using correlation-based metrics able to identify different types of relationships between data. They have long been identified as an adequate fitness metric in SR applications related to data modeling [40]. Regarding knowledge discovery, we selected various metrics with interesting characteristics: Pearson correlation which can be used to tackle linear relationships between variables; Spearman correlation which has the ability to uncover non-linear relationships. From these two metrics, some new fitnesses have been declined. For instance, we tried to restrict the population to positively correlated individuals by setting down to zeros all individuals with negative fitness. The modified correlation score Sc is then defined as $Sc(y, yhat) = max(0, corr(y, yhat))$ with $corr$ the chosen correlation metric. This new score won't actually restrict the search space as we took care of inserting a subtraction operator in the grammar to transform negative correlations into positive ones easily. Nevertheless, it prevents the algorithm from evolving two distinct populations alongside respectively made of anti-correlated and positively-correlated individuals.

To further prevent the bloating phenomenon, we adjusted the fitness function by adding a penalty term, similarly to what is done in [41]. In this study, they define a penalty term also called "Simplicity score" $Sy = \frac{max\_nodes - 0.5 * nb\_nodes - 0.5}{max\_nodes - 1}$ where max_nodes is the maximum number of nodes allowed in an individual and nb_node the actual number of nodes in the individual. This score ranges between and 0.5 and 1.

A score of 1 means that the expression has the smallest possible size and only contains one node, while a score of 0.5 is obtained if the individual's size reaches the maximum number of allowed nodes. They decided to set the lower limit to 0.5 to penalize large-sized individuals without forcing them to disappear as they can carry partially good genetic material. max_nodes is then a critical parameter to calibrate as a too-large value tends to create large individuals while a too-small one would restrict too much the search space.

In order to have a stronger penalty for individuals with a large number of nodes, without penalizing too much the small individuals, we chose to adjust the proposed score as follows : $Sy' = \frac{max\_nodes^{pow} - 0.5 * (nb\_nodes^{pow} + 1)}{max\_nodes^{pow} - 1}, pow \in \mathbb{N}$. The parameter pow is initialized with a low value of 3 but could be modified by the user (as in Figure 4 zone D) to constraint the complexity of the solution. Eventually, the fitness then becomes : $fitness = Sc * Sy'$.

4.4. Implementation

As backbone code, we upgraded the open-source implementation of GP in Python PonyGE2 [42]. This framework provides an efficient parallel implementation useful on large datasets such as those we use in our experiments. To propose up-to-date technical solutions, we improved many elements from initialization functions to the search engine. Especially, we inserted correlation-based error-metrics, an evolutionary step with population filtering, and we also extended the handling of grammar to probabilistic grammars.

5. Interactive platform

In this work, we want to take advantage of expert knowledge by interacting with an expert as the final user. To achieve this goal, we propose a web-based platform built using Plotly and Dash frameworks in Python. The platform has two elements: first, the web interface on the client-side where the user interacts with the algorithm directly in the browser, then, the core training performed on the server-side using the algorithm design we described earlier in Section 4. The user is solicited at four key stages of the algorithm :

- before launching the training, for hyperparameter calibration
- at initialization, to include user-defined individuals
- in between specific iterations to propose new individuals based on the analysis of top-ranked individuals from the previous generation

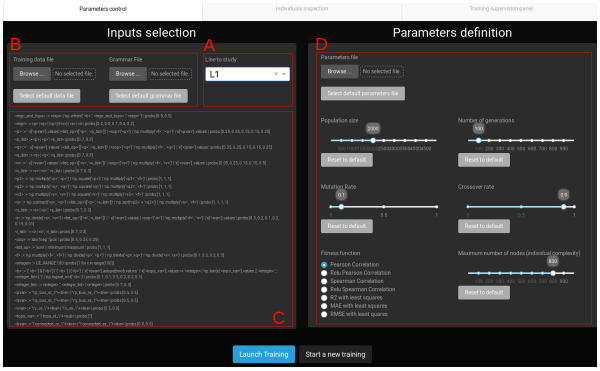

Figure 3. Parameters control tab. Mandatory target selection is in zone A. Zones B and C are dedicated to the definition (optional) of a custom grammar and data set. Other learning parameters of the PG3P algorithm can also be changed in zone D.

- at the end of each run, to draw some conclusions from the experiments regarding both the quality of the solutions and the relevance of selected hyperparameters.

To give multiple degrees of freedom to the users in the way they interact with the algorithm, we propose a web interface separated into three tabs: first, a parameters control tab to specify hyperparameters before launching the training; then, an inspection tab where the user can analyze the current population and provide feedback used in the next generations; and eventually a supervision tab that allows following the evolution of specific learning-related parameters from one generation to the next.

5.1. Parameters control tab

The first action the user performs is hyperparameters calibration, performed in the interface presented in Figure 3. For domain experts, it consists of at least defining the target y we focus on (zone A). Dataset and grammar can either be loaded or set a predefined default file (zone B). In zone C, the user can also view the default grammar. He can then modify and improve it in a new file, which is to be downloaded in zone B. ML experts can additionally set up GP parameters such as mutation and crossover rates, population size, number of generations, the fitness function, and the maximum number of nodes an individual can have (zone D).

5.2. Inspection and feedback tab

Interactivity definition Once the learning starts, the user is inquired to propose custom individuals as starting points before the first crossover and mutation operations.

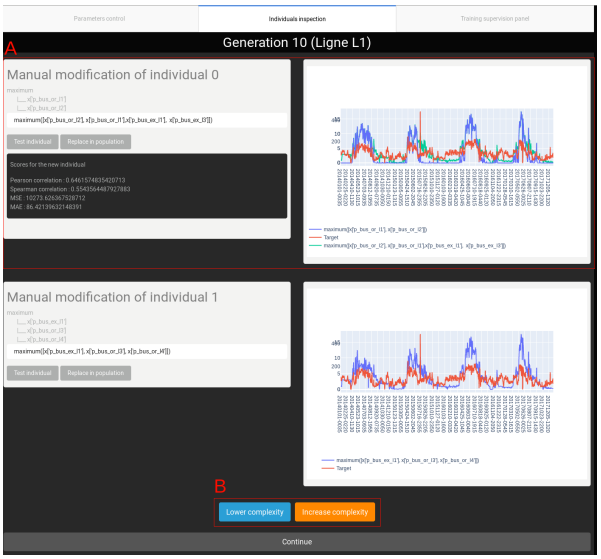

Figure 4. Inspection tab (top section) for individuals analysis. User can inspect, update, test and visualize individuals in zone A. Buttons in zone B allows the expert to increase or decrease the expressions maximal complexity in the next generations.

Figure 5. Inspection tab (bottom section) for further analysis. Zone C compares several expressions over time against the target. In zone D, the user can select other expressions to analyze, using their scores, among top-10 + 10-randomly-selected individuals.

The corresponding interactive visualization is displayed in Figures 4 and 5. Figure 5 should be contiguous and placed below Figure 4. This step is quite important as GP algorithms are known to be sensitive to initialization. This inspection-feedback step is also proposed similarly after a fixed number of iterations and performed using the same tab. This visualization shows the individuals from the current generation (or individuals created using PI_grow strategy if still at the initialization stage), ranked by their fitness value.

At the top (zone A in Figure 4), users can visualize the best individuals selected in the bottom-array (zone D in Figure 5). On the right, a temporal visualization provides a qualitative result regarding the quality of the solution. On the left, the individual is represented in a human-readable fashion using a tree-based representation with functions as nodes and terminal variables as leaves. Under this representation, the user can type in custom solutions as text to compare them to the current individual. The corresponding scores are evaluated and computed below. Once the user is satisfied with this new individual, he can insert it into the current population so that it will be used to generate the next generation. In zone C (Figure 5), another temporal graph compares all selected individuals' global behavior to the target variable. After inspection, and only if necessary, the user can either increase or decrease the complexity parameter and then proceed with the next generation (zone B in Figure 4)

At the bottom of the page (zone D), a summary of the generation shows the individuals' expression ranked by fitness and several other metrics such as R2, Mean Square Error, Mean Absolute Error. The expressions are shifted and rescaled using least-squared regression before evaluating the metric to have a relevant comparison between individuals. By default, only the two best individuals are selected for manual inspection, but the user can manually select any other individual.

At the end of the training, the final population is displayed similarly. From these interactions, the user can draw some conclusions about how the training went in order to upgrade the grammar rules and probabilities or change the selected fitness in the next experiments.

Regarding users' fatigue One of the issues to tackle in IE is the limitation of user fatigue. In our platform, we propose to select in the overall population the top-10 individuals and add ten additional individuals taken at random in the rest of the population for diversity purposes. Another strategy we proposed to minimize the users' fatigue is to perform manual inspection only at specific generations (by default, one in ten). This approach also seems to be relevant for real-world experiments because an expert's time is expensive, and the less interaction is required, the more likely it is that the experts will have enough time to test our approach.

5.3. Supervision tab

This last tab (Figure 6) displays general statistics about the training. On the top part (A), two line-charts give information about the fitness' evolution throughout the training while at the bottom (B), four charts provide

Figure 6. Training supervision tab. Zone A describes the scores evolutions while zone B give insights on the evolution of the population complexity.

information about the complexity of individuals in the population. On each graph, the statistics of the current run are statically superimposed on that of non-interactive training. Non-interactive statistics were obtained beforehand on 40 evolutionary runs.

These graphs allow for in-depth critics of the training procedure. It can be used to update the training parameters in the next runs, such as calibrating the complexity-constrain in the fitness or performing early stopping to prevent too complex individuals.

6. Usage scenarios

This section describes two scenarios for using our platform: the first one, where we do not get good results in the non-interactive case, and the second, where we want to refine a solution that already has a high fitness score. Rather than describing the detailed scores on the real-world dataset we embed, we focus on a description of high-level tasks to improve feature search.

6.1. Knowledge distillation on bad performing power lines

We first focus on the case where the PG3P alone is not sufficient to build up a robust solution. To exemplify how the user can take advantage of the platform, we compare in Figure 7 interactive and non-interactive setups on a line where the best fitness was under 0.8 without interactivity. The blue box plot depicts the evolution of the fitness at each generation on 40 non-interactive runs, and the red line-plot shows the best

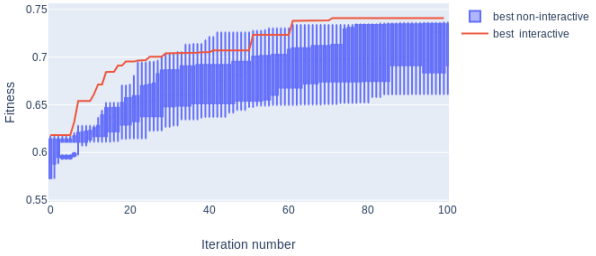

Figure 7. Comparison between interactive and 40 non-interactive runs on a challenging target variable. Non-interactive runs are represented by a blue box plot and the interactive run by a red line-chart.

fitness during an interactive run with an expert user.

While several standard (non-interactive) runs tend to reach a plateau and a local optimum after a few iterations and improve very little after the 50^{th} generation, the interactive run is still refining the model after the 70^{th} run. These elements advocate the benefit of interactivity to avoid local optima. Here, it might also be beneficial to increase the number of generations for the algorithm to converge. Moreover, the interactive version achieves the best results from the 60^{th} generation onwards. Because the expert instilled knowledge from the early generations, the algorithm can reach the same fitness score earlier. However, because the interactive-run does not outperform the non-interactive one in the first generations, we can infer that the expert can't distill his knowledge in the too-simple individuals (exhibited at the beginning) or that the user and the algorithm need some iterations to calibrate.

Eventually, we foresee that custom in-run complexity calibrations, e.g., increasing the maximally allowed complexity of individuals progressively, can also improve the search's overall performance. However, further experiments are necessary to show it.

6.2. Improving sparsity on already successful solutions

The second situation we address is one where a standard non-interactive evolution already gives good results with respect to the fitness score. In this case, non-interactive runs converge in a few iterations to a solution close to the optimal solution, and the algorithm proposes solutions with very similar scores but various complexities. Expert's knowledge would be useful here to identify the most relevant fitness-sparsity trade-off.

Figure 8 shows a example of such a situation. In this case, two similarly good solutions are displayed where they almost perfectly match the target visually. The first one, in red, is the most complex but most accurate,

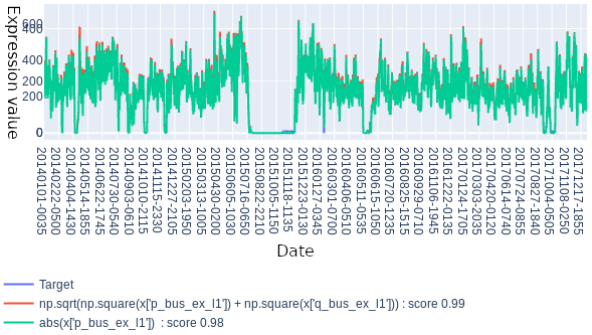

Figure 8. Temporal view of individuals with varying complexity, on a simplistic case. Both solutions (red and green lines) visually match the target variable (in blue) and have a similar score. Domain expertise is required to identify the most relevant solution.

while the green solution is a simplification. The question asked here is: Is the simplest solution sufficient or too extreme? In this situation, only an expert could tell which solution to prefer by referring to its historical knowledge. A strategy we propose to identify the most simple but accurate solution is to progressively increase or reduce the allowed complexity during the training to obtain both the simplified solution and the most precise one in the same population. The same strategy could also be used to prevent overfitting by manually removing all non-informative parameters in the expression.

7. Conclusion and perspectives

In this paper, we propose a HiTL approach for the understanding of physical phenomena. We describe a multi-level interactive platform designed for expert knowledge elicitation. It is a system designed for real-life application - in this case, power network monitoring - where experts can provide insights about their understanding of the problem at various stages of the process, both within each run and between two runs. We also propose two interaction scenarios corresponding to both sides of the spectrum, and we show how an expert can improve the search results.

This work opens to numerous perspectives. First, the interaction types we describe here are prototypical and could be enhanced, for example, by adding a user-centered metric filled in by the expert to take into account its knowledge and judgment even further. This new metric could then be either handled using a multi-objective strategy or integrated into fitness formulation. We also envision improving the probabilities update at each step according to the representativeness of productions at each iteration [43].

In addition, given that an expert's time is expensive, we were not able until then to make extensive user studies. Our very next step will then be to build an experience with a panel of junior expert users in training. Finally, even if the proposed platform only handles PG3P on the server-side for now, we foresee that our framework could also be helpful to interact with other types of algorithms as a core computation scheme.

References

[1] H. Haes Alhelou, M. E. Hamedani-Golshan, T. C. Njenda, and P. Siano, "A survey on power system blackout and cascading events: Research motivations and challenges," *Energies*, vol. 12, no. 4, 2019.

[2] B. Donnot, I. Guyon, M. Schoenauer, P. Panciatici, and A. Marot, "Introducing machine learning for power system operation support," in *IREP Symposium*, 2017.

[3] B. McKay, M. J. Willis, and G. W. Barton, "Using a tree structured genetic algorithm to perform symbolic regression," in *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 487–492, 1995.

[4] J. R. Koza, "Hierarchical automatic function definition in genetic programming," in *Proceedings of the Second Workshop on Foundations of Genetic Algorithms.*, pp. 297–318, 1992.

[5] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. 2011.

[6] V. Bachu and J. Anuradha, "A review of feature selection and its methods," *Cybernetics and Information Technologies*, vol. 19, p. 3, 03 2019.

[7] I. T. Jolliffe, *Principal Component Analysis*. Springer Series in Statistics, 1986.

[8] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[9] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[11] D. A. Augusto and H. J. C. Barbosa, "Symbolic regression via genetic programming," in *Proceedings. Vol.1. Sixth Brazilian Symposium on Neural Networks*, pp. 173–178, 2000.

[12] M. Virgolin, T. Alderliesten, A. Bel, C. Witteveen, and P. A. Bosman, "Symbolic regression and feature construction with gp-gomea applied to radiotherapy dose reconstruction of childhood cancer survivors," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1395–1402, 2018.

[13] A. Murari, E. Peluso, M. Lungaroni, M. Gelfusa, and P. Gaudio, "Application of symbolic regression to the derivation of scaling laws for tokamak energy confinement time in terms of dimensionless quantities," *Nuclear Fusion*, vol. 56, p. 026005, dec 2015.

[14] S. Sahoo, C. Lampert, and G. Martius, "Learning equations for extrapolation and control," vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 4442–4450, 10–15 Jul 2018.

[15] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Čeperić, and M. Soljačić, "Integration of neural network-based symbolic regression in deep learning for scientific discovery," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2020.

[16] A. Anjum, F. Sun, L. Wang, and J. Orchard, "A novel neural network-based symbolic regression method: Neuro-encoded expression programming," in *International Conference on Artificial Neural Networks*, pp. 373–386, 2019.

[17] B. K. Petersen, M. Landajuela, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," in *International Conference on Learning Representations*, 2021.

[18] D. Lynch, J. McDermott, and M. O'Neill, "Program synthesis in a continuous space using grammars and variational autoencoders," in *PPSN (2)*, vol. 12270 of *Lecture Notes in Computer Science*, pp. 33–47, 2020.

[19] M. L. Wong and K. Leung, "An induction system that learns programs in different programming languages using genetic programming and logic grammars," in *ICTAI*, pp. 380–387, 1995.

[20] D. J. Montana, "Strongly typed genetic programming," *Evolutionary computation*, vol. 3, no. 2, pp. 199–230, 1995.

[21] M. Keijzer and V. Babovic, "Dimensionally aware genetic programming," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, vol. 2, pp. 1069–1076, 01 1999.

[22] S. Prieschl, D. Girardi, and G. Kronberger, "Using ontologies to express prior knowledge for genetic programming," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pp. 362–376, 2019.

[23] D. E. Knuth, "backus normal form vs. backus naur form," *Commun. ACM*, vol. 7, no. 12, pp. 735–736, 1964.

[24] A. Ratle and M. Sebag, "Genetic programming and domain knowledge: Beyond the limitations of grammar-guided machine discovery," in *Parallel Problem Solving from Nature, 6th International Conference*, vol. 1917, pp. 211–220, 2000.

[25] M. Schoenauer and M. Sebag, "Using domain knowledge in evolutionary system identification," *CoRR*, vol. abs/cs/0602021, 2006.

[26] N. Lourenço, J. M. Colmenar, J. I. Hidalgo, and O. Garnica, "Structured grammatical evolution for glucose prediction in diabetic patients," in *GECCO*, pp. 1250–1257, 2019.

[27] N. Cherrier, J. Poli, M. Defurne, and F. Sabatié, "Consistent feature construction with constrained genetic programming for experimental physics," in *IEEE Congress on Evolutionary Computation, CEC*, pp. 1650–1658, 2019.

[28] J. J. Dudley and P. O. Kristensson, "A review of user interface design for interactive machine learning," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 8, no. 2, pp. 1–37, 2018.

[29] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, *et al.*, "Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems," *arXiv preprint arXiv:1903.12394*, 2019.

[30] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *Ai Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

[31] A. Kosorukoff, "Human based genetic algorithm," in *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace*, vol. 5, pp. 3464–3469 vol.5, 2001.

[32] J. Lv, M. Zhu, W. Pan, and X. Liu, "Interactive genetic algorithm oriented toward the novel design of traditional patterns," *Inf.*, vol. 10, no. 2, p. 36, 2019.

[33] Y. Yang and X. Tian, "Combining users' cognition noise with interactive genetic algorithms and trapezoidal fuzzy numbers for product color design," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1019749:1–1019749:11, 2019.

[34] J. T. Kim, S. Kim, and B. K. Petersen, "An interactive visualization platform for deep symbolic regression," in *IJCAI*, pp. 5261–5263, 2020.

[35] S. Wang and H. Takagi, "Improving the performance of predicting users' subjective evaluation characteristics to reduce their fatigue in iec.," *Journal of physiological anthropology and applied human science*, vol. 24 1, pp. 81–5, 2005.

[36] Kamalian, Ying Zhang, Takagi, and Agogino, "Reduced human fatigue interactive evolutionary computation for micromachine design," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 9, pp. 5666–5671 Vol. 9, 2005.

[37] A. Marot, S. Tazi, B. Donnot, and P. Panciatici, "Guided machine learning for power grid segmentation," in *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pp. 1–6, 2018.

[38] A. Ratle and M. Sebag, "Avoiding the bloat with stochastic grammar-based genetic programming," in *International Conference on Artificial Evolution (Evolution Artificielle)*, pp. 255–266, 2001.

[39] C. Ryan, J. J. Collins, and M. O'Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *Genetic Programming, First European Workshop, EuroGP'98, Proceedings*, vol. 1391, pp. 83–96, 1998.

[40] D. A. Savic, G. A. Walters, and J. W. Davidson, "A genetic programming approach to rainfall-runoff modelling," *Water resources management*, vol. 13, no. 3, pp. 219–231, 1999.

[41] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, and E. L. Michalkiewicz, "A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets," *Artificial Intelligence in Medicine*, vol. 30, no. 1, pp. 27–48, 2004.

[42] M. Fenton, J. McDermott, D. Fagan, and et al., "Ponyge2: grammatical evolution in python," in *GECCO (Companion)*, pp. 1194–1201, 2017.

[43] L. F. D. P. Sotto and V. V. de Melo, "A probabilistic linear genetic programming with stochastic context-free grammar for solving symbolic regression problems," in *GECCO*, pp. 1017–1024, 2017.