Genetic Algorithm with Linkage Learning

Anonymous Author(s)

ABSTRACT

Next-generation genetic algorithms (GAs) should explore information from the problem structure whenever possible. Variable interactions can be inferred using linkage learning. Statistical linkage learning techniques were shown to improve GAs' effectiveness significantly in many problems, but may eventually report false linkages. On the other hand, empirical linkage learning (ELL) techniques discover only true variable dependencies. However, traditional ELL techniques are computationally expensive. We introduce the genetic algorithm with linkage learning (GAwLL), which discovers an empirical weighted variable interaction graph (VIGw) as a side-effect of the optimization performed by a GA, making it a no-cost ELL technique. Vertices of the VIGw represent decision variables and weights indicate the strength of the interaction between variables. The VIGw allows us to obtain new insights about the optimization problem and can be used to design genetic operators that efficiently explore the information about variable dependencies. Experiments with NK landscapes show that GAwLL is able to efficiently build the empirical VIGw. We also present an interesting machine learning application, where the VIGw represents a feature interaction network. By using GAwLL, the feature interaction network is built as a side-effect of evolutionary feature selection.

CCS CONCEPTS

• **Mathematics of computing** → **Combinatorial optimization**;
• **Theory of computation** → **Random search heuristics**.

KEYWORDS

Genetic Algorithms, Variable Interaction Graph, Linkage Learning

ACM Reference Format:
Anonymous Author(s). 2023. Genetic Algorithm with Linkage Learning. In *Proceedings of The Genetic and Evolutionary Computation Conference 2023 (GECCO '23).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Genetic algorithms (GAs) have powerful features that make them very attractive for optimization applications. Using a population of individuals allows mixing them to potentially discover higher quality solutions. Populations also allow the use of solution diversity to improve exploration and offer opportunities for massive parallelization. However, when compared to other heuristics and metaheuristics, the performance of traditional GAs may sometimes be disappointing. In order to make genetic algorithms more competitive for specific problems, Whitley [19] proposes that next-generation GAs should be built using information from the problem structure whenever possible. Recently, some transformation operators and search strategies that explore problem structure have been developed, which can significantly improve the performance of evolutionary algorithms [1, 2, 16].

Information about the interaction between variables can be efficiently stored and manipulated in a *variable interaction graph* (VIG). The VIG is an undirected graph where vertices are related to decision variables and edges indicate nonlinear interactions between variables [1]. The VIG is available *a priori* in gray-box optimization problems, such as MAX-SAT and NK-Landscapes, by inspecting the fitness function components. In black-box optimization, an empirical VIG can be estimated by using linkage learning. The interaction between variables can be inferred before or during the optimization process [13]. Some state-of-the-art optimizers for combinatorial problems are based on *statistical linkage learning* (SLL) [5, 7, 14], where the frequencies of variable value combinations are counted. On the other hand, *empirical linkage learning* (ELL) techniques [11, 12] are based on analyzing the differences in the fitness evaluation when comparing neighboring solutions. ELL never reports a *false linkage*, i.e., a variable interaction that does not exist. However, traditional ELL techniques are computationally expensive, requiring many additional fitness evaluations for estimating the interactions between variables.

Tinós et al. [15] proposed *local search with linkage learning* (LSwLL) for iterated local search. LSwLL is a local search algorithm integrated with a new ELL technique based on *direct linkage empirical discovery* (DLED)[12]. LSwLL builds an empirical VIG as a side effect of optimization by local search. Unlike DLED and previous ELL techniques, no additional fitness evaluations are necessary for finding the interactions between variables. LSwLL was extended in LSwLL2 [10]. LSwLL2 automatically discovers an empirical *weighted variable interaction graph* (VIGw) as a side-effect of optimization by local search. The VIGw is a weighted graph, where weights indicate the strength of the interaction between variables. The VIGw provides new insights about the optimization problem and allows designing transformation operators that efficiently explore information about variables' interaction.

We propose the *genetic algorithm with linkage learning* (GAwLL). GAwLL discovers an empirical VIGw as a side-effect of the optimization performed by a GA. GAwLL builds an empirical VIGw during the execution of the GA without requiring additional fitness evaluations in order to estimate the interaction between variables. The VIGw is introduced in Section 2. GAwLL is presented in Section 3. In Section 4, we describe an interesting machine learning application of GAwLL, where the VIGw is used for visualizing feature interaction, i.e., the VIGw is a feature interaction network. By using GAwLL, the feature interaction network is built as a side-effect of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '23, July 15–19, 2023, Lisbon, Portugal
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

evolutionary feature selection performed by a GA. Results of experiments with NK landscapes and feature selection are presented and analyzed in Section 5.

2 THE WEIGHTED INTERACTION GRAPH

In this paper we focus on pseudo-Boolean optimization problems, $f : \mathbb{B}^N \rightarrow \mathbb{R}$, where N is the dimension of the candidate solution $\mathbf{x} = \{x_0, x_1, \ldots, x_{N-1}\}$, and $\mathbb{B} = \{0, 1\}$ and $\mathbb{R}$ are respectively the binary and real domains. An important aspect of problem structure is the nonlinear interaction between decision variables. Informally, we can say that there is an interaction between decision variables x_g and x_h if the impact, on $f(\mathbf{x})$, of changing x_h depends on the value of x_g. If the impact on $f(\mathbf{x})$ of changing x_h does not depend on the values of x_g, then we can say that the variables do not interact.

Formally, we define the dependence between variables by using the *Walsh decomposition* [6]. Any pseudo-Boolean function can be written as a weighted sum:

$$f(\mathbf{x}) = \sum_{i=0}^{2^N-1} w_i \psi_i(\mathbf{x}) \tag{1}$$

where $w_i \in \mathbb{R}$ is the i-th Walsh coefficient, $\psi_i(\mathbf{x}) = (-1)^{\mathbf{i}^T\mathbf{x}}$ generates a sign, and $\mathbf{i} \in \mathbb{B}^N$ is the binary representation of index i. By using the Walsh decomposition, the interaction between variables is defined as follows [15].

Definition 2.1. Given a pseudo-Boolean function $f : \mathbb{B}^N \rightarrow \mathbb{R}$, we say that variables x_g and x_h *interact* in $f(\mathbf{x})$ if there exists at least one nonzero Walsh coefficient w_i in the Walsh decomposition of $f(\mathbf{x})$ such that the g-th and h-th elements of $\mathbf{i}$ are equal to one. The *variable interaction* relationship is symmetric.

Information about the interaction between decision variables in $f(\mathbf{x})$ can be efficiently stored and manipulated using a VIG, defined as follows.

Definition 2.2. The *variable interaction graph* (VIG) is an undirected graph $G = (V_G, E_G)$, where each vertex $v_i \in V_G$ is related to a decision variable x_i, and each edge $(v_g, v_h) \in E_G$ indicates that variables x_g and x_h interact in $f(\mathbf{x})$ (Definition 2.1).

The weighted VIG, denoted by VIGw [10], is an extension of the VIG. Given a candidate solution $\mathbf{x} \in \mathbb{B}^N$, we can define the following fitness differences:

$$\delta_g(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{1}_g) - f(\mathbf{x}) \tag{2}$$

and

$$\delta_g(\mathbf{x} \oplus \mathbf{1}_h) = f\big(\mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_g\big) - f(\mathbf{x} \oplus \mathbf{1}_h) \tag{3}$$

where $\oplus$ is the XOR bitwise operation, $\mathbf{1}_g \in \mathbb{B}^N$ denotes a characteristic vector with the g-th element equal to one and all other elements equal to zero, $f(\mathbf{x} \oplus \mathbf{1}_g)$ denotes the fitness of solution $\mathbf{x}$ after flipping x_g, and $f\big(\mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_g\big)$ is the fitness of solution $\mathbf{x}$ after flipping x_h and x_g.

The strength of an interaction and the VIGw are defined based on the following equation:

$$\omega_{g,h}(\mathbf{x}) = |\delta_g(\mathbf{x} \oplus \mathbf{1}_h) - \delta_g(\mathbf{x})|. \tag{4}$$

Definition 2.3. Given a pseudo-Boolean function $f : \mathbb{B}^N \rightarrow \mathbb{R}$, the *strength* of the interaction between variables x_g and x_h in $f(\mathbf{x})$ (Definition 2.1) is given by:

$$v(g, h) = \frac{1}{2^N} \sum_{\mathbf{x} \in \mathbb{B}^N} \omega_{g,h}(\mathbf{x}) \tag{5}$$

where $\omega_{g,h}(\mathbf{x})$ is given by Eq. (4). Strength is symmetric, i.e., $v(h, g) = v(g, h)$.

Definition 2.4. The *weighted variable interaction graph* (VIGw) is an undirected weighted graph $G = (V_G, E_G, W_G)$, where each vertex $v_g \in V_G$ is related to a decision variable x_g, and each edge $(v_g, v_h) \in E_G$ with nonzero weight $v(g, h) \in W_G$ indicates that variables x_g and x_h interact in $f(\mathbf{x})$ with strength $v(g, h)$ (Definition 2.3).

Computing $v(g, h)$ has exponential worst-case time complexity for arbitrary functions. In addition, the problem structure must be known *a priori* to do so. We can use an *empirical* VIGw instead of the (*true*) VIGw.

Definition 2.5. Let $G = (V_G, E_G, W_G)$ be the VIGw (Definition 2.4) for $f(\mathbf{x})$. An *empirical VIGw*, $G_p = (V_{G_p}, E_{G_p}, W_{G_p})$, is a graph where $V_{G_p} = V_G$, $E_{G_p} \subset E_G$, and $\hat{v}(g, h) \in W_{G_p}$ is the weight associated to edge (v_g, v_h), given by:

$$\hat{v}(g, h) = \frac{1}{|\Upsilon(g, h)|} \sum_{\mathbf{x} \in \Upsilon(g,h)} \omega_{g,h}(\mathbf{x}) \tag{6}$$

where $\Upsilon(g, h) \subseteq \mathbb{B}^N$ is the subset of candidate solutions visited to compute $\omega_{g,h}(\mathbf{x})$.

3 GA WITH LINKAGE LEARNING (GAWLL)

Tinós et al. [15] show that variables x_g and x_h interact in $f(\mathbf{x})$, according to Definition 2.1, and (v_g, v_h) is an edge of the VIG if $\omega_{g,h}(\mathbf{x}) > 0$ for any $\mathbf{x} \in \mathbb{B}^N$. LSwLL [15] flips decision variables of candidate solutions in a specific order to detect edges of the VIG. LSwLL2 [10] builds an empirical VIGw instead of an empirical VIG where the strength of the interaction between variables is given by Eq. (4). To check if $\omega_{g,h}(\mathbf{x}) > 0$, LSwLL and LSwLL2 compute the fitness of three solutions generated from current solution $\mathbf{x}$: i) solution $\mathbf{x}$ with x_g flipped, i.e., $\mathbf{x} \oplus \mathbf{1}_g$; ii) solution $\mathbf{x}$ with x_h flipped, i.e., $\mathbf{x} \oplus \mathbf{1}_h$; iii) solution $\mathbf{x}$ with x_g and x_h flipped, i.e., $\mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_h$. LSwLL and LSwLL2 do this in local search changing the order in which variables are visited in standard first improvement local search.

We propose to check if $\omega_{g,h}(\mathbf{x}) > 0$ in solutions generated by 1-bit and 2-bit flip mutation in a GA. Alg. 1 shows the pseudo-code of the *generation* procedure in the standard GA. In Alg. 1: $\mathbf{P}$ is the parent population, $\mathbf{Q}$ is the new population, $|\mathbf{P}|$ is the population size, $\mathbf{x}, \mathbf{y} \in \mathbf{P}$ are two parents chosen by selection, p_c is the crossover rate, and p_m is the mutation rate. Note that the *generation* procedure of the standard GA was re-organized in Alg. 1 to highlight that part of the new population is generated by crossover and mutation and part is generated only by mutation. For simplicity, we do not include solutions generated by elitism in Alg. 1.

GAwLL differs from the standard GA mainly in the way solutions are generated by mutation alone. We call this procedure *mutation with linkage learning*. In mutation with linkage learning, three solutions are generated from a parent $\mathbf{x} \in \mathbf{P}$. The first two solutions

Algorithm 1 Generation of standard GA

1: $i = 1$
2: **while** $i \leq |\mathbf{P}|$ **do**
3: **if** $i \leq |\mathbf{P}| - 1$ **and** rand()$\leq p_c$ **then**
4: $(\mathbf{x}, \mathbf{y})$=selection($\mathbf{P}$)
5: $(Q(i), Q(i+1))$=crossover($\mathbf{x}, \mathbf{y}$)
6: $Q(i)$=mutation($Q(i), p_m$)
7: $Q(i+1)$=mutation($Q(i+1), p_m$)
8: evaluate $Q(i)$ and $Q(i+1)$
9: $i = i + 2$
10: **else**
11: $\mathbf{x}$=selection($\mathbf{P}$)
12: $Q(i)$=mutation($\mathbf{x}, p_m$)
13: evaluate $Q(i)$
14: $i = i + 1$
15: **end if**
16: **end while**

are equal to $\mathbf{x}$, with exception of flipping x_g (first offspring) or x_h (second offspring). The last offspring is equal to $\mathbf{x}$ with x_g and x_h flipped. In this way, we can compute $\omega_{g,h}(\mathbf{x})$ (Eq. 4) and check if variables x_g and x_h interact. We can also compute the strength of the interaction between x_g and x_h (Eq. 6).

GAwLL builds a weighted graph G_p along the generations. Initially, the graph G_p has one vertex for each decision variable and no edges. The *generation* procedure of GAwLL adds edges to G_p as a side effect of creating a new population. Alg. 2 shows the pseudo-code of the *generation* procedure in GAwLL. There are two main differences compared to the standard GA: i) a deterministic number of offspring is generated by crossover and mutation (line 3). While standard GA generates an averaged number of offspring by crossover and mutation equal to $p_c|\mathbf{P}|$, GAwLL always generates $2\lceil p_c|\mathbf{P}|/2 \rceil$ offspring by crossover and mutation, where [.] indicates round to nearest integer; ii) in steps 11-25 (mutation with linkage learning), solutions generated only by mutation have a fixed number of bit flips in GAwLL, while in the standard GA, a random number of bit flips, with rate p_m, occur. It is important to observe that no additional fitness evaluations are needed in GAwLL in order to find the edges of graph G_p and compute the strenght of the interactions. Figures 1 and 2 show graph G_p in two examples of mutation with linkage learning. The following theorem shows that the graph G_p returned by GAwLL is an empirical VIGw and, as a consequence, it returns no false linkage.

Theorem 3.1. *The weighted graph G_p returned by GAwLL is an empirical VIGw (Definition 2.5) of $f(\mathbf{x})$.*

Proof. In order to show that the graph $G_p = (V_{G_p}, E_{G_p}, W_{G_p})$ returned by GAwLL is an empirical VIGw (Definition 2.5), we need fist to compare it with the VIGw $G = (V_G, E_G, W_G)$ for the problem's instance. Given a pseudo-Boolean function $f : \mathbb{B}^N \rightarrow \mathbb{R}$, we know that the number of decision variables is N. The graph G_p is initially created with one vertex for each decision variable and with no edges. Then, $V_{G_p} = V_G = \{v_0, v_1, \ldots, v_{N-1}\}$, that is one condition for an empirical VIGw (Definition 2.5).

Edges (v_g, v_h) are added to E_{G_p} only by mutation with linkage learning along the generations (Alg. 2). The condition $\omega_{g,h}(\mathbf{x}) > 0$ is tested in mutation with linkage learning only after generating three offspring from parent $\mathbf{x}$: $Q(i) = \mathbf{x} \oplus \mathbf{1}_g$; $Q(i+1) = \mathbf{x} \oplus \mathbf{1}_h$;

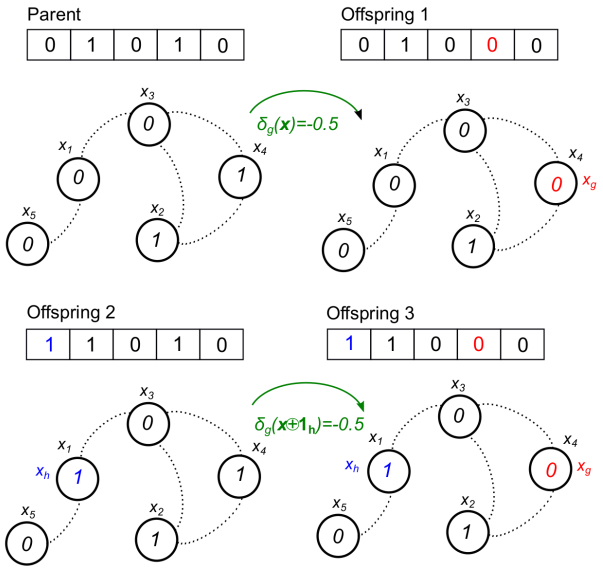

Figure 1: In this example, 3 offspring are generated from mutation with linkage learning. Variables x_1 and x_4 are respectively mutated in offspring 1 and 2. Both variables are mutated in offspring 3. In this example, no interaction is detected between x_1 and x_4.

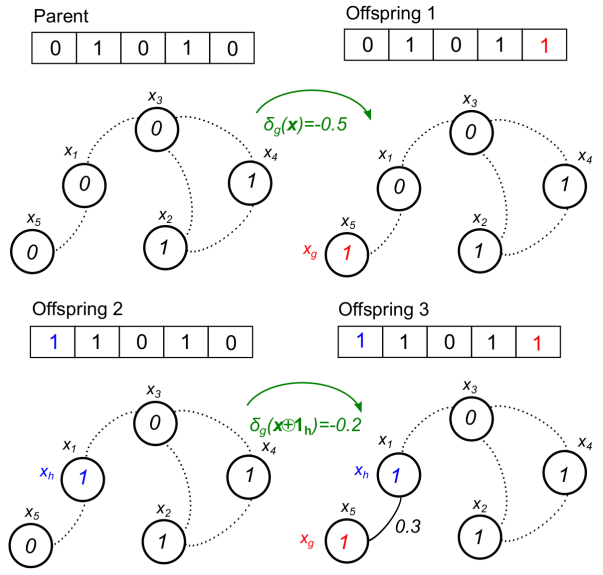

Figure 2: Variables x_1 and x_5 are respectively mutated in offspring 1 and 2. Both variables are mutated in offspring 3. In this example, interaction between x_1 and x_5 is detected. As a consequence, an edge between vertices v_1 and v_5, with weight 0.3, is added to graph E_{G_p}.

and $Q(i+2) = \mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_g$. The proof that an edge $(v_g, v_h) \in E_{G_p}$ found by mutation with linkage learning is also an edge of VIGw E_G is based on the Walsh decomposition of $f(\mathbf{x})$. Given a candidate

Algorithm 2 Generation of GAwLL

```
1: i = 1
2: while i ≤ |P| do
3:    if i ≤ |P| − 1 and i ≤ p_c|P| then
4:       (x, y)=selection(P)
5:       (Q(i), Q(i+1))=crossover(x, y)
6:       Q(i)=mutation(Q(i), p_m)
7:       Q(i+1)=mutation(Q(i+1), p_m)
8:       evaluate Q(i) and Q(i+1)
9:       i = i + 2
10:   else if i ≤ |P| − 2 then
11:      x=selection(P)
12:      Q(i) = Q(i+1) = Q(i+2) = x
13:      randomly choose h and g
14:      flip x_g in Q(i)
15:      flip x_h in Q(i+1)
16:      flip x_g and x_h in Q(i+2)
17:      evaluate Q(i), Q(i+1), and Q(i+2)
18:      if ω_{g,h}(x) > 0 then
19:         if edge (v_g, v_h) ∈ E_{G_p} then
20:            update weight of edge (v_g, v_h) by using Eq. (6)
21:         else
22:            add edge (v_g, v_h), with weight ω_{g,h}(x)
23:         end if
24:      end if
25:      i = i + 3
26:   else
27:      x=selection(P)
28:      Q(i)=mutation(x, p_m)
29:      evaluate Q(i)
30:      i = i + 1
31:   end if
32: end while
```

solution $\mathbf{x}$, Eq. (1) can be rewritten as follows:

$$f(\mathbf{x}) = \sum_{i \in C} f_i(\mathbf{x}) \tag{7}$$

where $f_i(\mathbf{x}) = w_i \psi_i(\mathbf{x})$ and $C \subset \{0, 1, \ldots, 2^N - 1\}$ contains only indices of components with nonzero Walsh coefficients. We can decompose Eq. (7) as:

$$f(\mathbf{x}) = \sum_{i \in C_{g,h}} f_i(\mathbf{x}) + \sum_{i \in C_g - C_{g,h}} f_i(\mathbf{x}) + \sum_{i \in C_h - C_{g,h}} f_i(\mathbf{x}) + R(\mathbf{x})$$

where: C_g is the subset indicating all components of the sum given by Eq. (7) that depend on variable x_g; C_h is the subset indicating all components that depend on variable x_h; and $C_{g,h} = C_g \cap C_h$ is the subset indicating all components that depend on both variables x_g and x_h. Finally:

$$R(\mathbf{x}) = \sum_{i \in C - (C_g \cup C_h)} f_i(\mathbf{x})$$

sums over all of the remaining terms that do not depend on x_g or x_h. $R(\mathbf{x})$ does not change when x_g or x_h change. Thus, the fitness of offspring $Q(i)$, $Q(i+1)$, and $Q(i+2)$ can be respectively written by:

$$f(\mathbf{x} \oplus \mathbf{1}_g) = \sum_{\substack{i \in \\ C_{g,h}}} f_i(\mathbf{x} \oplus \mathbf{1}_g) + \sum_{\substack{i \in \\ C_g - C_{g,h}}} f_i(\mathbf{x} \oplus \mathbf{1}_g) + \sum_{\substack{i \in C_h - \\ C_{g,h}}} f_i(\mathbf{x}) + R(\mathbf{x}),$$

$$f(\mathbf{x} \oplus \mathbf{1}_h) = \sum_{\substack{i \in \\ C_{g,h}}} f_i(\mathbf{x} \oplus \mathbf{1}_h) + \sum_{\substack{i \in \\ C_g - C_{g,h}}} f_i(\mathbf{x}) + \sum_{\substack{i \in C_h - \\ C_{g,h}}} f_i(\mathbf{x} \oplus \mathbf{1}_h) + R(\mathbf{x}),$$

$$f(\mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_g) = \sum_{\substack{i \in \\ C_{g,h}}} f_i(\mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_g) + \sum_{\substack{i \in C_g - \\ C_{g,h}}} f_i(\mathbf{x} \oplus \mathbf{1}_g) + \sum_{\substack{i \in C_h - \\ C_{g,h}}} f_i(\mathbf{x} \oplus \mathbf{1}_h) + R(\mathbf{x}).$$

Thus, from equations (4), (2), (3):

$$\omega_{g,h}(\mathbf{x}) = \left|\left(f(\mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_g) - f(\mathbf{x} \oplus \mathbf{1}_h)\right) - \left(f(\mathbf{x} \oplus \mathbf{1}_g) - f(\mathbf{x})\right)\right| = \sum_{i \in C_{g,h}} \left|f_i(\mathbf{x} \oplus \mathbf{1}_h \oplus \mathbf{1}_g) - f_i(\mathbf{x} \oplus \mathbf{1}_h) - f_i(\mathbf{x} \oplus \mathbf{1}_g) + f_i(\mathbf{x})\right|$$

that is greater than zero only if $C_{g,h} \neq \emptyset$. As a consequence, $v(g,h)$ (Eq. 5) and $\hat{v}(g,h)$ (Eq. 6) are higher than zero only when $C_{g,h} \neq \emptyset$. In other words, if $\omega_{g,h}(\mathbf{x}) > 0$ holds, then variables x_g and x_h interact in $f(\mathbf{x})$, and there exists at least one nonzero Walsh coefficient w_i in the Walsh decomposition of $f(\mathbf{x})$ such that the g-th and h-th elements of $\mathbf{i}$ are both equal to one (Definition 2.1). Thus, an edge $(v_g, v_h) \in E_{G_p}$ found by mutation with linkage learning is also an edge of VIGw E_G, i.e., $E_{G_p} \subset E_G$, that is the second condition in Definition 2.5. It follows that GAwLL never returns a false linkage. Finally, $\hat{v}(g,h)$ is computed in Alg. 2 by using Eq. (6), which is the third condition in Definition 2.5. In Alg. 2, $\Upsilon(g,h) \subseteq \mathbb{B}^N$ is the subset of visited solutions where condition $\omega_{g,h}(\mathbf{x}) > 0$ holds. □

Proposition 3.2. *The number of edges of the empirical VIGw G_p created by running GAwLL, with generation procedure given by Alg. 2, for n_g generations is bounded by:*

$$0 \leq |E_{G_p}| \leq \min\left(n_g \lfloor n_m/3 \rfloor, |E_G|\right) \tag{8}$$

where $|E_G|$ is the number of edges of the respective VIGw G and $n_m = |\mathbf{P}| - 2\left[p_c|\mathbf{P}|/2\right]$.

Proof. From Theorem 3.1, we know that $E_{G_p} \subset E_G$. If $|E_G| = 0$ or condition $\omega_{g,h}(\mathbf{x}) > 0$ does not hold for all mutations with linkage learning, then $|E_{G_p}| = 0$. The number of individuals generated by crossover and mutation in each generation of GAwLL is $2\left[p_c|\mathbf{P}|/2\right]$. Thus, the number of individuals generated only by mutation in each generation of GAwLL (Alg. 2) is:

$$n_m = |\mathbf{P}| - 2\left[p_c|\mathbf{P}|/2\right].$$

The number of offspring generated by mutation with linkage learning in each generation is:

$$n_l = 3\lfloor n_m/3 \rfloor.$$

In order to test condition $\omega_{g,h}(\mathbf{x}) > 0$ and detect a possible interaction between x_g and x_h, we need to generate 3 offspring: $Q(i)$, $Q(i+1)$, and $Q(i+2)$. Then, the maximum number of edges of the VIGw that we discover in each generation is:

$$n_e = n_l/3 = \lfloor n_m/3 \rfloor.$$

Thus, after n_g generations:

$$|E_{G_p}| \leq \min(n_g \lfloor n_m/3 \rfloor, |E_G|).$$

□

4 FEATURE SELECTION AND INTERACTION

Feature selection is an important task in machine learning and data mining [9]. Many real-world datasets have a large number of features and some of them can be redundant or irrelevant for the given machine learning task. In many applications, the specialist do not know in advance the most relevant features. Feature selection can be used for finding the most relevant features in these applications. Given a dataset, there are two main feature selection approaches: filter and wrapper. Filter algorithms do not depend on machine learning models, i.e., features are selected based only on the statistical properties of the dataset. On the other hand, wrapper algorithms are optimizers that use a given machine learning model to evaluate subsets of features.

Wrapper feature selection based on evolutionary algorithms has received increasing attention from machine learning community in the last decades [20]. By using GAs, (wrapper) feature selection can be viewed as a pseudo-Boolean optimization problem. Given a dataset and a machine learning model, a candidate solution $\mathbf{x} \in \mathbb{B}^N$ indicates a subset of selected features, where N is the number of features in the dataset and $x_i = 1$ if the i-th feature is selected and $x_i = 0$ otherwise. Different metrics can be used for evaluating the fitness $f(\mathbf{x})$, e.g., accuracy or F1 score for classification and sum of the squared errors for regression.

Feature importance, or *variable importance*, is also a relevant task that has been receiving a lot of attention from the machine learning community [8]. Feature selection is a very simple model for finding the importance of features. However, more sophisticated models exist. These models rank the variable importance according to their relevance for the machine learning task. For example, feature importance can be estimated by measuring the performance degradation of a trained model after data permutation.

Much less attention has been given to another relevant task in machine learning: *feature interaction*, or *variable interaction*. In general, it is much harder to estimate feature interaction than to estimate feature importance. Variable interaction can be defined as a scalar quantity that indicates the degree to which two or more input variables combine to affect the output variable [8]. In another definition, Friedman and Popescu [4] say that two variables, x_h and x_g, interact if the impact on the output function $f(\mathbf{x})$ of changing x_h depends on x_g. Based on this definition, Friedman and Popescu [4] propose the *H-statistic* or *H-index*. The H-statistic for a pair of variables measures the change in the predicted output of a machine learning model as variables vary over their marginal distribution.

Inglis et al. [8] proposed to use *H-statistic* to obtain a *feature interaction network*. In this network, vertices represent variables and weighted edges indicate the strength of the interaction between variables. Detecting variable interaction, by using the H-statistic, in [4, 8] is similar to what is done in the GAwLL and earlier in the LSwLL2 [10]. In GAwLL, feature interaction between variables x_h and x_g is indicated by the weight of edge (g, h) of the empirical VIGw. Finding feature interaction is obtained as a side-effect of using GAwLL for feature selection. In other words, no additional fitness evaluation is needed; the H-statistic needs many evaluations of a machine learning model to compute the variable interaction for each pair of variables. By using GAwLL, the empirical VIGw can be employed for the visualization of all two-variable interactions, i.e., the VIGw is a feature interaction network.

5 EXPERIMENTS

GAwLL builds a empirical VIGw during optimization and differs from the standard GA in the way mutation without crossover is performed. In Section 5.1, we compare GAwLL with the standard GA to check how mutation with linkage learning and the additional operations needed for estimating the VIGw impact the optimization performed by the GA. In the experiments, the GAs run for a fixed time. In GAwLL and standard GA, tournament selection, elitism, population restart, bit-flip mutation, and uniform crossover are used. The parameters are: crossover rate $p_c = 0.6$, mutation rate $p_m = \frac{1}{N}$, where N is the size of the binary string, population size $|P| = 100$, and size of the tournament selection poll equal to 3. Population restart replaces all individuals of the current population, except the best one, by random individuals. The initial population is also random. In addition, we present results for the experiments where first improvement local search optimizes all individuals in the initial population and after restarts. If local search is not used, population restart is applied when the fitness of the best individual does not change for 50 generations. If local search is used, population restart is applied every 50 generations.

Two measures, computed for each run of the GAs, are used when comparing the algorithms: **FIT**: fitness of the best solution found by the GA; and **GEN**: number of generations of the GA. A workstation with processor Intel@ Core i7-12700 Alder Lake (12 cores, 20 threads, 25 MB cache) and 128 GB of RAM was used for running the GAs implemented in C++[1].

GAwLL and standard GA were applied to two optimization problems: NK landscapes and feature selection. We present results of experiments with adjacent and random NK landscapes for different dimensions, N, and epistasis degree, $k = K + 1$. Five instances were generated for each model and combination of N and k. The number of runs for each instance was ten. The runtime for each GA was fixed in $\frac{Nk}{5}$ seconds.

The GAs do not use *a priori* information about the structure of the problems, i.e., the problems are considered as black-box optimization problems. In fact, we have information about the structure of the problem in NK landscapes; this information is not available in the feature selection problem considered here. However, knowing the structure of the problem in NK landscapes is important because we want to check the ability of GAwLL in discovering it.

In feature selection, the machine learning model is the *K-nearest neighbors* (KNN) algorithm, with $K = 3$. The evaluation function is:

$$f(\mathbf{x}) = 0.98 f_1(\mathbf{x}) + 0.02 \frac{N - \sum_{i=0}^{N-1} x_i}{N} \tag{9}$$

where $f_1(\mathbf{x})$ is a measure for the performance of the machine learning model and the second term depends on the number of selected features. In classification, $f_1(\mathbf{x})$ is the rate of test set examples correctly classified by the KNN with features indicated by $\mathbf{x}$. In regression, $f_1(\mathbf{x}) = 1 - E(\mathbf{x})$, where $E(\mathbf{x})$ is the mean squared error for the test set when the KNN with features indicated by $\mathbf{x}$ is used. Here, the sizes of the training and test sets were respectively $0.7n_{ex}$

[1]The source code for GAwLL will be freely available on GitHub at (omitted).

and $0.3n_{ex}$, where n_{ex} is the number of examples (samples) in the dataset. Results for 5 datasets are presented. The datasets, listed in Table 1, are from the UCI Machine Learning Repository [3], except for dataset covidxr [10], that was obtained by extracting radiomic features from chest x-ray images. The images were selected at random from the COVIDx training dataset [18], a public dataset containing x-ray images of three classes: normal, and patients with COVID19 and non-COVID19 pneumonia. By using PyRadiomics [17], 93 texture features were extracted from the images. The number of runs for each dataset was 10. The fixed runtime for the GAs was $\frac{Nn_{ex}}{10}$ seconds.

Table 1: Datasets in the feature selection problem.

dataset	type	features (N)	examples (n_{ex})	outputs
housing	regression	13	506	1
zoo	classification	16	101	7
ionosphere	classification	34	351	2
sonar	classification	60	208	2
covidxr	classification	93	456	3

In Section 5.2, the empirical VIGws generated by GAwLL are analysed. When comparing the empirical VIGw to the VIGw in the NK landscapes over the generations, the GAs without local search run for a fixed number of generations, instead of a fixed runtime.

5.1 Results: performance

Tables 2 and 3 show the results for the GAs with and without local search. The median for best fitness (FIT) and number of generations (GEN) is presented. Results of the Wilcoxon signed rank test with significance level 0.01 are presented for statistically comparing the results of GAwLL and (standard) GA. When comparing *FIT* for GAwLL and GA, the null hypothesis (no statistical difference between the results) cannot be rejected in 22 out of 24 experiments of NK landscapes and in all 10 experiments of feature selection.

When local search was employed, the GA presented better results more times, but statistical difference between these results was observed in only 1 out of 18 experiments. When local search was not employed, the GA presented better results for 2 out of 5 feature selection instances, but worse results for most of the NK landscapes instances. But, again, statistical difference between the results was observed in only 1 out of 18 experiments. These results can be explained by the number of generations (GEN) for each GA. The GAs presented better FIT generally when presented better GEN. Most of the results of GEN are statistically significant. However, statistically better results for GEN did not imply in statistically better results for FIT. GA presented better GEN results in the experiments with feature selection. GAwLL presented better GEN results in the experiments with NK landscapes for the algorithms without local search. The empirical VIGw had much more edges in the experiments with features selection than in the experiments with NK landscapes. In addition, evaluating the fitness required more time in feature selection. Anyway, the results are dependent on the choice of the parameters of the GAs, especially the mutation and crossover rates.

Table 2: Median of FIT and GEN for the NK landscapes for experiments with and without local search. The symbols '=', '+', and '–' respectively indicate that the median for the GAwLL is equal, better or worse than the median of (standard) GA. The Wilcoxon signed rank test was employed to statistically compare the results of GAwLL and GA. The letter *s* indicates that the null hypothesis (no statistical difference between the results) can be rejected according to the significance level.

				without local search		with local search	
measure	model	N	k	GA	GAwLL	GA	GAwLL
FIT	adjacent	100	3	0.74999	0.75133(+)	0.75155	0.75125(-)
			5	0.76448	0.76440(-)	0.76746	0.76746(-)
		500	3	0.74036	0.74053(+)	0.73813	0.73560(-)
			5	0.75041	0.75403(s+)	0.74456	0.74250(-)
		1000	3	0.73639	0.73785(+)	0.72471	0.72424(-)
			5	0.74602	0.74748(+)	0.72848	0.72799(-)
	random	100	3	0.74644	0.74513(-)	0.75380	0.75380(=)
			5	0.76041	0.76276(+)	0.78416	0.78613(+)
		500	3	0.73918	0.74077(+)	0.73942	0.73879(-)
			5	0.75355	0.75206(-)	0.75536	0.75471(-)
		1000	3	0.73532	0.73549(+)	0.72862	0.72809(-)
			5	0.74794	0.74977(+)	0.74555	0.74445(s-)
GEN	adjacent	100	3	140603	151414(s+)	24072	24174(s+)
			5	159035	166006(s+)	25704	25959(s+)
		500	3	147798	158199(s+)	4743	4743(s=)
			5	170526	177421(s+)	4667	4692(+)
		1000	3	153187	164129(s+)	2142	2142(=)
			5	172311	178731(s+)	2193	2193(=)
	random	100	3	140625	151323(s+)	22287	22542(s+)
			5	160306	163914(s+)	21012	20978(-)
		500	3	147723	158191(s+)	4131	4174(s+)
			5	170893	177644(s+)	3468	3468(s=)
		1000	3	153932	164211(s+)	1734	1683(-)
			5	171427	178535(s+)	1530	1530(=)

Table 3: Median of FIT and GEN for the feature selection problem for experiments with and without local search.

		without local search		with local search	
measure	dataset	GA	GAwLL	GA	GAwLL
FIT	housing	0.98665	0.98665(=)	0.98665	0.98665(=)
	zoo	0.90016	0.90016(=)	0.90016	0.90016(=)
	ionosphere	0.96873	0.96815(-)	0.96932	0.96932(=)
	sonar	0.99533	0.99517(-)	0.99567	0.99567(=)
	covidxr	0.79608	0.80302(+)	0.78317	0.78850(+)
GEN	housing	1127	1069(s-)	663	629(s-)
	zoo	7519	6522(s-)	4284	3897(s-)
	ionosphere	2364	2044(s-)	561	557(-)
	sonar	3311	2853(s-)	559	510(s-)
	covidxr	1191	1135(-)	153	144(-)

5.2 Results: VIGw

GAwLL never returns a false linkage (Theorem 3.1), i.e., if two vertices are connected in the empirical VIGw, they are also connected in the VIGw and in the VIG. In NK landscapes, the VIG is known *a priori*. Figure 3 shows the percentage of edges of the VIGw G in the empirical VIGw G_p over generations for one run of GAwLL for each combination of models and parameters of Nk landscapes.

Table 4 shows the percentage of the edges of the VIGw found by GAwLL in the empirical VIGw for the experiments with NK landscapes presented in Section 5.1. In the experiments without local search, GAwLL discovered at least a median of 99.8% of the edges of the VIGw. In the experiments with local search, GAwLL discovered a median of 100% of the edges of the VIGw for NK

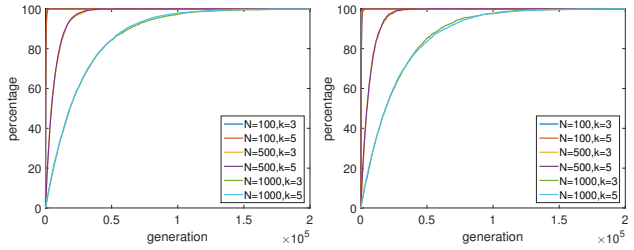

Figure 3: Percentage of edges of empirical VIGw in the VIGw for random (left) and adjacent (right) NK landscapes. In these experiments, GAwLL without local search run for a fixed number of generations.

Table 4: Median of the percentage of the edges of the VIGw in the empirical VIGw found by GAwLL for the NK landscapes experiments with and without local search.

model	N	k	without local search	with local search
adjacent	100	3	100.0	100.0
		5	100.0	100.0
	500	3	100.0	51.6
		5	100.0	50.3
	1000	3	99.8	7.8
		5	99.9	7.9
random	100	3	100.0	100.0
		5	100.0	100.0
	500	3	100.0	46.9
		5	100.0	41.1
	1000	3	99.8	6.3
		5	99.9	5.6

landscapes with $N = 100$. However, the percentages was only 5.6% for $N = 1,000$ and $k = 5$. These results are explained by the number of generations (Proposition 3.2), showed in Table 2, and by the fact that the VIGw has more edges for higher N and k.

Figure 4 shows an example of empirical VIGw for adjacent NK landscapes with $N = 30$ and $k = 3$. The fitness function $f(\mathbf{x})$ in the NK landscapes is composed of N subfunctions $f_i(\mathbf{x})$, each one dependent on k variables of $\mathbf{x}$. In the adjacent NK landscapes model, subfunction $f_i(\mathbf{x})$ depend on x_i and in $k-1$ adjacent variables. In the *standard* adjacent model, the contribution of each subfunction to $f(\mathbf{x})$ is a random number uniformly generated in the range $[0.0, 1.0]$. In the experiment here (Figure 4), the contribution of each subfunction to $f(\mathbf{x})$ is a also a random number, but uniformly generated in different ranges. For the first 3 subfunctions, the contributions are generated in the range $[0.0, 1.0]$. The contributions of the next 3 subfunctions are generated in the range $[0.0, 0.9]$. Then, for the next 3 subfunctions, the contributions are generated in the range $[0.0, 0.8]$, and so on. The range for the last 3 subfunctions is $[0.0, 0.1]$. We call this model as *distributed* adjacent NK landscapes. This example show how the VIGw can be much more informative than the VIG. In the empirical VIGw presented in Figure 4, the weights for edges connecting vertices related to higher contributions of $f_i(\mathbf{x})$ were generally higher. In other words, the interaction strength were higher for pairs of variables that, when changed, impacted more the fitness function. This information helps understanding the problems and the algorithms. It can also help designing new reproduction operators.

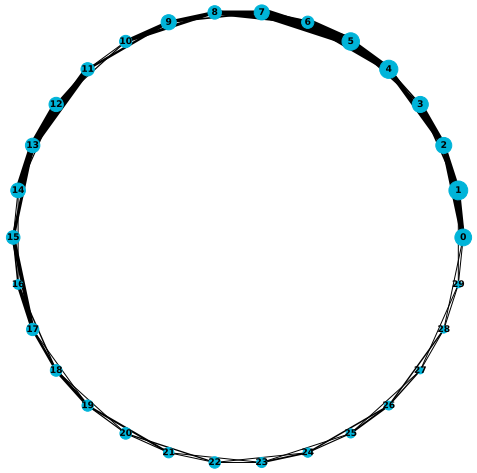

Figure 4: Example of empirical VIGw found by GAwLL for the distributed adjacent NK landscapes with $N = 30$ and $k = 3$. The widths of the lines are proportional to the weights of the respective edges. The size of the i-th node is proportional to the contribution of subfunction $f_i(\mathbf{x})$ for the evaluation of the best individual found by the GAwLL with local search.

Figure 5 shows the VIGws obtained in the first run of GAwLL without local search applied to 4 instances of the experiments with feature selection presented in Section 5.1. The VIGws obtained in the feature selection experiments are dense graphs. In this way, only the edges with the largest weights of the empirical VIGw are shown in Figure 5; the largest weights are selected according to the procedure presented in [10]. In the housing dataset, the features selected by GAwLL were: NOX, RM, LSTAT, and TAX. Using a different machine learning model (RuleFIT), the feature importance algorithm proposed by Friedman and Popescu [4] indicated NOX, RM, and LSTAT as the features with highest relative importance. TAX was the 6th variable with highest relative importance. Figure 5 indicates that the strongest interactions are related to LSTAT; a similar result was obtained by Friedman and Popescu [4]. LSwLL2 also obtained a similar result too [10]. The VIGww for dataset zoo, ionosphere, and covidxr are also presented in Figure 5. It is interesting to observe that the strongest interactions do not necessary occur for the features selected by GAwLL.

6 CONCLUSIONS

GAwLL builds an empirical VIGw as a side-effect of the optimization performed by a GA. Results of experiments with NK landscapes show that GAwLL builds empirical VIGws with 99.8% or more of the edges of the VIG. No false linkage is returned. In experiments with NK landscapes and features selection, no significant difference of the performance was observed when GAwLL is compared to standard GA with mutation rate $p_m = \frac{1}{N}$. However, these results are dependent on the parameters of the GAs. We show that the number of edges of the VIGw discovered by GAwLL depends on

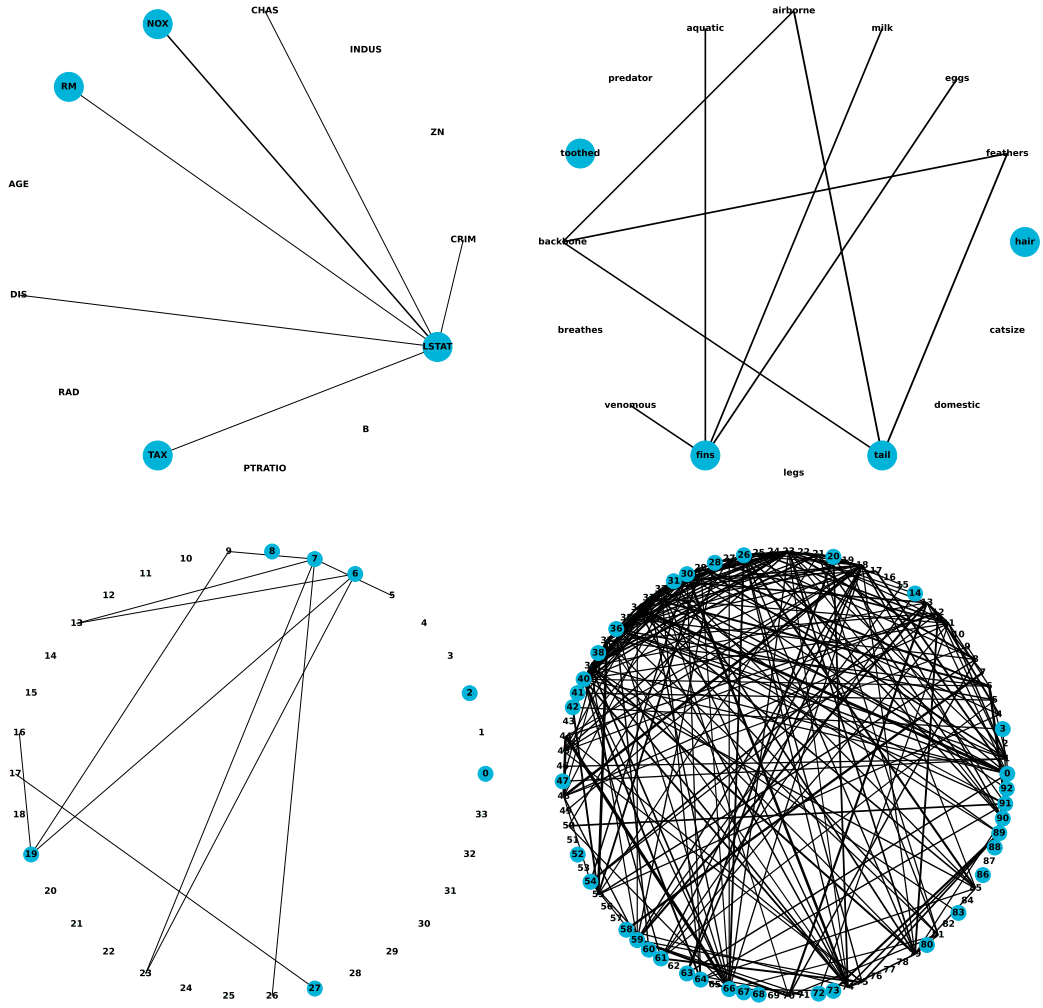

Figure 5: Empirical VIGws found in the first run of GAwLL for the feature selection problem with datasets: housing (top, left), zoo (top, right), ionosphere (botton, left), and covidxr (botton, right). The widths of the lines are proportional to the weights of the respective edges. The features selected by GAwLL are indicated by the blue circles. Only the edges with largest weights are presented for each graph.

parameters of the GA (number of generations, crossover rate, and size of the population) and of the problem (dimension and number of edges of the VIGw).

The VIGw produced by GAwLL allows us to obtain new insights about the optimization problem and algorithms. The strongest interactions occur for pairs of variables that when changed most impact evaluation function $f(\mathbf{x})$. In the experiments with NK landscapes, pairs of variables associated to subfunctions with largest contribution to the evaluation function $f(\mathbf{x})$ generally presented the strong connections, i.e., the largest weights of the empirical VIGw. Experiments with feature selection showed that the empirical VIGw can be used for the visualization of networks of interactions between features in machine learning. Understanding feature interaction is a relevant problem in machine learning. Feature interaction networks are produced by GAwLL during feature selection, without needing additional fitness evaluations.

Mutation with linkage leaning can be employed in other evolutionary algorithms in the future. GAwLL can be integrated with gray-box optimization operators and strategies, e.g., partition crossover[16] and *deterministic recombination and iterated local search* (DRILS) [1], for black-box optimization. In the NK landscapes experiments, GAwLL with local search discovered few edges of the VIGw for instances with high dimension and high epistasis. In these cases, GAwLL can be combined with LSwLL2 in a hybrid GA with local search.

REFERENCES

[1] F. Chicano, G. Ochoa, D. Whitley, and R. Tinós. 2022. Dynastic potential crossover operator. *Evolutionary Computation* 30, 3 (2022), 409–446.

[2] K. Deb and C. Myburgh. 2016. Breaking the billion-variable barrier in real-world optimization using a customized evolutionary algorithm. In *Proc. of GECCO'2016*. 653–660.

[3] D. Dua and C. Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[4] J. H. Friedman and B. E. Popescu. 2008. Predictive learning via rule ensembles. *The Annals of Applied Statistics* (2008), 916–954.

[5] B. W. Goldman and W. F. Punch. 2014. Parameter-less Population Pyramid. In *Proc. of GECCO'2014*. 785–792.

[6] R. B. Heckendorn. 2002. Embedded Landscapes. *Evolutionary Computation* 10, 4 (2002), 345–369.

[7] S.-H. Hsu and T.-L. Yu. 2015. Optimization by Pairwise Linkage Detection, Incremental Linkage Set, and Restricted / Back Mixing: DSMGA-II. In *Proc. of GECCO'2015*. 519–526.

[8] A. Inglis, A. Parnell, and C. B. Hurley. 2022. Visualizing variable importance and variable interaction effects in machine learning models. *Journal of Computational and Graphical Statistics* (2022), 1–13.

[9] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. 2017. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)* 50, 6 (2017), 1–45.

[10] Omitted. 2023. (2023).

[11] M. W. Przewozniczek and M. M. Komarnicki. 2020. Empirical Linkage Learning. *IEEE Transactions on Evolutionary Computation* 24, 6 (2020), 1097–1111.

[12] M. W. Przewozniczek, M. M. Komarnicki, and B. Frej. 2021. Direct linkage discovery with empirical linkage learning. In *Proc. of GECCO'2021*. 609–617.

[13] D. Thierens and P. A. N. Bosman. 2012. Predetermined versus Learned Linkage Models. In *Proc. of GECCO'2012*. 289–296.

[14] D. Thierens and P. A. N. Bosman. 2013. Hierarchical problem solving with the linkage tree genetic algorithm. In *Proc. of GECCO'2013*. 877–884.

[15] R. Tinós, M. W. Przewozniczek, and D. Whitley. 2022. Iterated local search with perturbation based on variables interaction for pseudo-boolean optimization. In *Proc. of GECCO'2022*. 296–304.

[16] R. Tinós, D. Whitley, and F. Chicano. 2015. Partition crossover for pseudo-boolean optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. 137–149.

[17] J. J. M. Van Griethuysen, A. Fedorov, C. Parmar, A. Hosny, N. Aucoin, V. Narayan, R. G. H. Beets-Tan, J.-C. Fillion-Robin, S. Pieper, and H. J. W. L. Aerts. 2017. Computational radiomics system to decode the radiographic phenotype. *Cancer Research* 77, 21 (2017), e104–e107.

[18] L. Wang and A. Wong. 2020. COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest radiography images. *arXiv preprint arXiv:2003.09871* (2020).

[19] D. Whitley. 2019. Next generation genetic algorithms: a user's guide and tutorial. In *Handbook of Metaheuristics*. Springer, 245–274.

[20] B. Xue, M. Zhang, W. N. Browne, and X. Yao. 2016. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation* 20, 4 (2016), 606–626.