A novel surrogate-assisted evolutionary algorithm applied to partition-based ensemble learning

Dushatskiy, Arkadiy; Alderliesten, Tanja; Bosman, Peter A.N.

DOI
10.1145/3449639.3459306

Publication date
2021

Document Version
Final published version

Published in
GECCO 2021 - Proceedings of the 2021 Genetic and Evolutionary Computation Conference

A Novel Surrogate-assisted Evolutionary Algorithm Applied to Partition-based Ensemble Learning

Arkadiy Dushatskiy
Centrum Wiskunde & Informatica
Amsterdam, the Netherlands
arkadiy.dushatskiy@cwi.nl

Tanja Alderliesten
Leiden University Medical Center
Leiden, the Netherlands
t.alderliesten@lumc.nl

Peter A. N. Bosman
Centrum Wiskunde & Informatica
Amsterdam, the Netherlands
TU Delft
Delft, The Netherlands
peter.bosman@cwi.nl

ABSTRACT

We propose a novel surrogate-assisted Evolutionary Algorithm for solving expensive combinatorial optimization problems. We integrate a surrogate model, which is used for fitness value estimation, into a state-of-the-art P3-like variant of the Gene-Pool Optimal Mixing Algorithm (GOMEA) and adapt the resulting algorithm for solving non-binary combinatorial problems. We test the proposed algorithm on an ensemble learning problem. Ensembling several models is a common Machine Learning technique to achieve better performance. We consider ensembles of several models trained on disjoint subsets of a dataset. Finding the best dataset partitioning is naturally a combinatorial non-binary optimization problem. Fitness function evaluations can be extremely expensive if complex models, such as Deep Neural Networks, are used as learners in an ensemble. Therefore, the number of fitness function evaluations is typically limited, necessitating expensive optimization techniques. In our experiments we use five classification datasets from the OpenML-CC18 benchmark and Support-vector Machines as learners in an ensemble. The proposed algorithm demonstrates better performance than alternative approaches, including Bayesian optimization algorithms. It manages to find better solutions using just several thousand fitness function evaluations for an ensemble learning problem with up to 500 variables.

CCS CONCEPTS

• **Mathematics of computing** → **Evolutionary algorithms**;

KEYWORDS

Expensive combinatorial optimization, surrogate-assisted Evolutionary Algorithms, Ensemble Learning

ACM Reference Format:
Arkadiy Dushatskiy, Tanja Alderliesten, and Peter A. N. Bosman. 2021. A Novel Surrogate-assisted Evolutionary Algorithm Applied to Partition-based Ensemble Learning. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459306

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459306

1 INTRODUCTION

Expensive combinatorial optimization is a combinatorial optimization subfield, in which fitness function evaluations are (computationally) expensive. In contrast to the case of optimization problems with inexpensive function evaluations, in case of problems with expensive function evaluations, search algorithms are usually evaluated by their ability to find good solutions within a limited number of function evaluations rather than by a required number of evaluations to find the global optimum. Expensive optimization problems arise in various domains. One traditional application is simulation-based optimization in engineering design [26, 27, 31]. Combinatorial expensive optimization has recently achieved increased attention due to the popularity of the Neural Architecture Search (NAS) field [16, 21]. The goal in NAS is to find the best Neural Network architecture for a particular task. A function evaluation that corresponds to a partial [3] or end-to-end [32] Deep Neural Network (DNN) training procedure can be extremely expensive, taking up to several hours [23]. A related application is finding an ensemble of deep learning models through data partitioning, for instance to tackle the problem of data heterogeneity in medical image analysis [20, 29]. Here too, the efficiency of the search algorithm is extremely important as each function evaluation again requires training a DNN. Therefore, novel efficient algorithms for combinatorial expensive optimization are highly demanded.

Bayesian Optimization (BO) is a traditional approach to solving expensive optimization problems. The most common BO algorithms are based on Gaussian processes, aimed at real-valued variables. For their use in the combinatorial domain, they were shown to have caveats [22]. Consequently, novel BO algorithms were developed, designed to handle categorical variables. Replacing the Gaussian Process surrogate model with Random Forests [11] was proposed in Sequential Model-based Algorithm Configuration (SMAC) [25]. Another popular BO algorithm, which was shown to perform better than traditional Gaussian process-based BO in categorical and mixed domains, is the Tree-structured Parzen Estimator (TPE) [4] and its implementation in the Hyperopt package [5]. Recently, two BO algorithms with better performance than SMAC and Hyperopt were proposed: Bayesian Optimization of Combinatorial Structures (BOCS) [2] and Combinatorial Bayesian Optimization using the Graph Cartesian Product (COMBO) [30]. The main idea of both these algorithms is to explicitly model interactions between

variables. While BOCS takes into account only second-order interactions, COMBO has no limit on the order of interactions and builds a sparse Bayesian regression model over possible high-order interactions between variables. BOCS is limited to binary variables only, while COMBO can handle variables with higher cardinality. COMBO showed state-of-the-art performance on both common binary optimization benchmark problems such as Ising Spin-glass and MAXSAT and an example of NAS. However, this comes at a cost of extremely expensive computations that grows fast as the number of function evaluations increases for maintaining the surrogate model. For instance, only experiments with up to 300 function evaluations were feasible in the original paper [30].

Surrogate-assisted Evolutionary Algorithms (EAs) are an interesting alternative approach to solving expensive optimization problems. Using surrogate models to replace part of the function evaluations done by an EA is a natural way to reduce the number of evaluations while keeping the powerful search characteristics of EAs. Common surrogate models are polynomials, Kriging, Radial Basis Function Network (RBFN), or Support Vector Regression (SVR) [17]. However, surrogate-assisted EAs are not as widely used for combinatorial optimization problems as for real-valued ones. A recent approach for combinatorial optimization integrates a Convolutional Neural Network with an advanced model-based EA called Convolutional Neural Network Surrogate-assisted GOMEA (CS-GOMEA), but it is limited to binary problems [19].

Local Search (LS) is a well-known simple, yet often effective search algorithm. The main principle is to perform greedy search (only improvements are accepted) in the vicinity of a current solution, leading to quick convergence to a local optimum. It was recently shown that LS can be very efficient for combinatorial formulations of NAS, achieving competitive performance with state-of-the-art EAs and outperforming a commonly adopted baseline in NAS - Random Search (RS) [16, 34].

Ensembling is a common Machine Learning technique to improve performance. Usually, individual models are simple learners such as Decision Trees. A classic approach is Bagging [10], where several models are trained on randomly selected subsets of samples. An alternative approach is to train models on disjoint subsets of samples [1, 12]. In contrast to Bagging, with such an approach it is possible to train each model on homogeneous data, increasing chances of better performance. Finding the best partitioning then, however, is a potentially expensive optimization problem.

In this paper, we propose a novel surrogate-assisted EA based on a state-of-the-art P3-like variant of GOMEA. Unlike CS-GOMEA [19], it is not limited to binary problems, does not make assumptions about problem regularity (subfunctions of fixed size), and uses a more efficient population management scheme - Parameterless Population Pyramid (P3). To the best of our knowledge, this is the first time a surrogate model is integrated into a P3-like EA. We also propose a simple, yet shown to be efficient adaptive mechanism to control and balance the number of performed real and surrogate fitness evaluations. The introduced algorithm is applied to an ensembling problem defined on supervised classification datasets. Support-vector Machines (SVMs) are used as learners in the considered ensembles.

2 SEARCH PROBLEMS AND ALGORITHMS

2.1 Problem formulation

In general, we consider unconstrained combinatorial optimization problems with categorical variables. Potentially, the cardinality of problem variables differs. However, for simplicity of notation in this work, we assume that all variables have cardinality α, meaning that all variables have α possible values. Mathematically, the considered problems can be formulated as a global optimization problem $\max_{x \in \mathcal{D}} f(x)$ where $f(x) : \mathcal{D} \rightarrow \mathbb{R}$, $\mathcal{D} = \{0, \ldots, \alpha - 1\}^{\ell}$, ℓ is the number of variables, and α is the alphabet size.

2.2 Local Search

In contrast to RS, LS traverses the search space in an ordered, local manner, quickly converging to a local optimum. In a random restart scenario, LS starts from a randomly generated solution. We specifically consider first improvement neighbourhood search as our LS. A single iteration of LS consists of considering all variables in a random order, and assigning to each variable the value in its domain that corresponds the best fitness function value. Iterations are repeated until a solution is not changed. After that, a new random solution is generated and the search loop is repeated. The Random Restart Local Search pseudocode is provided in Algorithm 1.

Algorithm 1: Random Restart Local Search Algorithm.

```
Function LocalSearch():
    do
        improved = false
        s ← generateRandomSolution()
        do
            s.fitness ← evaluate(s)
            for i ∈ randomPermutation({0, ..., ℓ − 1}) do
                s' ← s
                for j ∈ {0, ..., α − 1} \ s_i do
                    s'_i ← j
                    s'.fitness ← evaluate(s')
                    if s'.fitness > s.fitness then
                        s_i ← s'_i
                        improved ← true
        while improved
    while budget is not exhausted
```

2.3 An adaptation of P3 for non-binary problems

We adapt the state-of-the-art model-based Evolutionary Algorithm known as the Parameterless Population Pyramid (P3) [24] for solving non-binary combinatorial optimization problems. P3 is a variant of the Gene-pool Optimal Mixing Algorithm (GOMEA) [8] that uses local search, a complementary donor search, and a fitness-pyramid structured growing population. We call the P3 variant used in this paper *P3-GOMEA*. Below, we give a brief overview of its features. Pseudocode is provided in Algorithm 2 and Algorithm 3.

2.3.1 Linkage Model. The idea of using a so-called Linkage Model for guiding the evolution process through variation restricted to specific variables and immediately testing for improvements was introduced with the concept of Optimal Mixing Evolutionary Algorithms (OMEAs) [33]. A Linkage Model is commonly defined in the form of a set of (possibly overlapping) subsets of variables.

Such a structure is called a Family Of Subsets (FOS). The goal is to use information about dependencies between variables to perform crossover more efficiently. In the case of Black-Box optimization these dependencies are not known a priori, and, therefore, they have to be learned during optimization. A much used approach is to first estimate pairwise dependencies from the population and build higher-order models upon this. For pairwise dependencies learning either Mutual Information (MI) [33] or Normalized Mutual Information (NMI) [24] is often used. Both MI and NMI calculations can be naturally extended to the case of non-binary variables [28]. P3-GOMEA uses the NMI measure. After pairwise dependencies are learned, they can be used for Linkage Model construction.

The FOS model known as the Linkage Tree (LT), which uses a hierarchical structure to store sets of dependent variables, was shown to often be the most efficient one [33]. An LT is a binary tree with $2\ell - 1$ nodes. LT leaves are singletons with variables. The root of an LT is a set containing all variables. All other nodes are linkage subsets F^i which are unions of disjoint subsets of children k, j of node i: $F^i = F^j \cup F^k, F^j \cap F^k = \emptyset$.

P3-GOMEA uses a filtered LT Model as proposed in [9], in which redundant subsets are removed. When two subsets F^j and F^k are merged into a subset F^i, it may happen that the MI or NMI measure between them is maximal (one). It is supposed that there is then no merit in using these subsets in variation separately as it may disrupt a building block F^i. Thus, keeping subsets F^j and F^k in an FOS is not reasonable and only the parent subset F^i is kept.

2.3.2 Gene-pool Optimal Mixing. The Gene-pool Optimal Mixing operator (GOM) is the variation operator used in the GOMEA family of algorithms. GOM is applied to a single solution and outputs a single solution that is never worse than the input solution. To improve a solution, GOM loops over the contents of the FOS model. For each linkage subset F^i, GOM attempts to overwrite the values of the variables in F^i of the solution in consideration, with values from a donor solution, often chosen at random from the population. If this overwriting action does not cause the fitness of the solution to become worse, the copy action is accepted. Otherwise, the donor material is rejected and the action is undone. The pseudocode of GOM is provided in Algorithm 3.

2.3.3 Parameterless population management scheme. Choosing an optimal population size is not trivial, and, therefore, a population management scheme without a population size parameter has much practical value. Several such schemes were proposed for the GOMEA family of algorithms [15]. In this paper, we consider P3 [24], which was shown to be efficient [15], is fully parameterless, and can be naturally adapted to a surrogate-assisted EA.

Solutions are stored in a pyramidal structure. Each layer of the pyramid is a set of solutions (duplicates are not stored). Linkage Models are learned separately for each pyramid level. In each iteration, only one solution is evolved. A new solution is generated randomly and added to the bottom layer of the pyramid. Then, by using solutions from the current pyramid level as donors, the current solution is evolved using GOM. If GOM leads to fitness improvement, the resulting solution is added to the next pyramid level, and GOM is performed at the next pyramid level. Otherwise, processing this solution is finished, and a new iteration starts with a new, randomly generated solution.

2.3.4 Hill Climber. Before evolving a solution using GOM, an LS algorithm (also called a Hill Climber in [24]) can be applied to quickly bring the solution to a local optimum. Such an approach is used in the original P3 algorithm [24]. However, our preliminary experiments showed that in the case of a limited number of allowable function evaluations (several thousand evaluations) such use of LS leads to similar performance as stand-alone LS (without P3) because the majority of function evaluations are spent on the LS phase. Therefore, we do not include LS in our P3-GOMEA.

Algorithm 2: P3-GOMEA and Surrogate-assisted P3-GOMEA (SA-P3-GOMEA). The lines which are added in the surrogate-assisted algorithm are highlighted in yellow.

Parameters: A relaxation parameter η
Function EA():
  $iter \leftarrow 0$
  $Pyramid \leftarrow [\emptyset]$
  $R \leftarrow \emptyset$
  for $i = 0, \ldots, \ell - 1$ **do**
    $R \leftarrow R \cup$ generateRandomSolution()
  trainSurrogateModel(R)
  $F \leftarrow$ predictSurrogateFitness(R)
  $\lambda \leftarrow 1$
  $\mathcal{T} \leftarrow$ setThreshold(F, λ)
  while $\neg terminationCriterionSatisfied$ **do**
    $elitist \leftarrow None$ // elitist value is updated if necessary when real evaluations are performed
    $p \leftarrow$ generateRandomSolution()
    $Pyramid^0 \leftarrow Pyramid^0 \cup \{p\}$
    $solutionsAdded \leftarrow$ true
    $currentTopLevel \leftarrow |Pyramid| - 1$
    $\mathcal{L} \leftarrow 0$
    $elitistBefore \leftarrow elitist$
    while $\mathcal{L} \leq currentTopLevel$ *and* $solutionsAdded$ **do**
      $\mathcal{F} \leftarrow$ learnLinkageModel($Pyramid^{\mathcal{L}}$)
      $o \leftarrow$ GOM($p, Pyramid^{\mathcal{L}}, \mathcal{F}$)
      if compareSolutions(o, p) **then**
        if $\mathcal{L} = currentTopLevel$ **then**
          $Pyramid^{\mathcal{L}+1}.append(\emptyset)$
        $Pyramid^{\mathcal{L}+1} \leftarrow Pyramid^{\mathcal{L}+1} \cup \{o\}$
        $solutionsAdded \leftarrow$ true
      $p \leftarrow o$
      $\mathcal{L} \leftarrow \mathcal{L} + 1$
    $elitistAfter \leftarrow elitist$
    if ¬compareSolutions(*elitistAfter, elitistBefore*) **then**
      $\lambda \leftarrow \lambda\eta$
    trainSurrogateModel(R)
    $F \leftarrow$ predictSurrogateFitness(R)
    $\mathcal{T} \leftarrow$ setThreshold(F, λ)

2.4 Surrogate-assisted EA

We integrate, in a novel way, a surrogate model into the above-described P3-GOMEA to replace a part of the real function evaluations with surrogate function evaluations. Our proposed approach to integrating surrogate models can be applied to various search algorithms. In this work, we focus on a surrogate-assisted version of the P3-GOMEA algorithm that we refer to as *SA-P3-GOMEA*.

Algorithm 3: Key functions used in the P3-GOMEA and SA-P3-GOMEA algorithms. The lines of code which are not used in the surrogate-assisted algorithm are highlighted in gray. The lines which are added in the surrogate-assisted algorithm are highlighted in yellow.

```
Variables: Real elitist solution elitist, threshold 𝒯, quantile λ, set of solutions
           with known real fitness values R
Function GOM(o, 𝒫, ℱ):
    backup ← o
    changed ← false
    ℱ ← sortFOSInAscendingOrder(ℱ)
    for i ∈ {0, 1, ..., |ℱ| − 1} do
        donorsList = randomPermutation({𝒫_0, 𝒫_1, ..., 𝒫_{n−1}})
        for j ∈ {0, 1, ..., n − 1} do
            d ← donorsList[j]
            o_Fi ← d_Fi
            if o_Fi ≠ d_Fi then
                evaluateSolution(o)
                if acceptChange (o) then
                    backup_Fi ← o_Fi
                    changed ← true
                else
                    o_Fi ← backup_Fi
                    break
    return o
Function evaluateSolution(x):
    x.fitness ← calculateFitness(x)
    x.realFitnessCalculated ← true
    x.surrogateFitness ← predictSurrogateFitness(x)
    if x.surrogateFitness > 𝒯 then
        x.fitness ← calculateFitness(x)
        x.realFitnessCalculated ← true
        R ← R ∪ {x}
    if x.realFitnessCalculated then
        if x.fitness > elitist.fitness then
            elitist ← x
            λ ← 1
Function compareSolutions(x,y):
    if x.realFitnessCalculated and y.realFitnessCalculated then
        if x.realFitness > y.realFitness then
            return 1;
        if x.realFitness == y.realFitness then
            return 0;
        if x.realFitness < y.realFitness then
            return -1;
    else
        if x.surrogateFitness > y.surrogateFitness then
            return 1;
        if x.surrogateFitness == y.surrogateFitness then
            return 0;
        if x.surrogateFitness < y.surrogateFitness then
            return -1;
```

The changes required to P3-GOMEA to obtain SA-P3-GOMEA are provided in Algorithm 2 and Algorithm 3.

2.4.1 Surrogate model integration. The outline of the proposed surrogate model integration into a search algorithm is as follows. First, some initial random solutions are evaluated. In general, the number of initial solutions can be tuned, but we use a simple approach that is also used in COMBO, namely, ℓ random solutions are generated and evaluated. Then, a surrogate model is trained on these solutions. Next, search is performed the same as without a surrogate model, but whenever solutions need to be evaluated, mainly surrogate (estimated by the surrogate model) evaluations are performed. Real function evaluations are performed only in case of high predicted fitness value. The rationale is that solutions with high surrogate fitness values should also have high real fitness values. However, for the sake of biasing the search competently toward high-quality solutions, we must be sure of just exactly how good this solution is.

The condition of performing a real function evaluation is defined as follows. Suppose, R is an array of solutions with known real fitness values and F is an array of corresponding surrogate fitness values. A real function evaluation is performed for a solution s, if its surrogate fitness f is greater than a threshold $\mathcal{T}$. This threshold is calculated as a λ-quantile value of the sorted values in array F. The value of λ is dynamically adjusted. In the beginning, $\lambda = 1$, meaning that a real function evaluation for a solution is performed only if its surrogate fitness exceeds the current surrogate elitist value. If an algorithm has not found new real elitists for a while, then we suppose that the current surrogate model is not accurate enough and it is reasonable to perform real evaluations more elaborately. Therefore, the value of λ is then multiplied by a hyperparameter η ($0 < \eta < 1$), effectively relaxing the condition for a real fitness evaluation. Once an algorithm finds a new elitist, λ is reset to 1. After evolving one solution in SA-P3-GOMEA, the surrogate model is retrained and values of F are re-calculated. The frequency of the model retraining can be potentially adjusted, but we stick to this natural retraining schedule. Note that the proposed surrogate-assisted EA has only one numeric hyperparameter η.

As a surrogate model is unlikely to predict surrogate fitness values with perfect precision, comparing real fitness values and surrogate fitness values is undesirable and leads to inferior performance [6, 19]. Therefore, for the comparison of the fitness values of two solutions the following strategy is used. If both solutions already have a real fitness value, they are compared via these. Otherwise, surrogate fitness values are calculated for both solutions and used for the comparison. The pseudocode of the comparison function is provided in Algorithm 3.

2.4.2 Surrogate model. Choosing the best surrogate model is often the key to good performance in surrogate-assisted optimization. However, the main goal of this paper is not to propose the best performing surrogate model, but rather to introduce a new adaptive mechanism for surrogate model integration, use it in P3-GOMEA, and test its performance on an ensembling problem. To underline the generality of our proposed mechanism, we experiment with four types of surrogate model.

(1) Multi-Layer Perceptron (MLP). In contrast to a Convolutional Neural Network used in [19], the considered Neural Networks are fully connected as we do not want to assume any regularity in input data such as subfunctions of fixed size. We use a fixed model structure with two hidden layers, each having the number of neurons equal to the input size.
(2) Decision Tree-based Gradient Boosting [13]. This type of model was shown to perform well on a variety of tasks [14, 18, 35] and can naturally handle categorical variables.

(3) Support Vector Regression (SVR). This is a common choice for surrogate-assisted EAs [17].
(4) Random Forest (RF).

To allow correct processing of categorical variables by MLP, RF and SVR, input solutions are one-hot encoded. Thus, if an optimization domain has ℓ variables with alphabet size α, surrogate models get samples of dimensionality $\ell\alpha$ as input.

Gradient Boosting, RF, and SVR have several hyperparameters such as kernel type for SVR, and learning rate for Gradient Boosting. Hyperparameters are tuned only once (before the first model training) using 3-fold cross-validation and simple grid search. We consider tuning the MLP architecture to be out of the scope of this paper. The list of tunable hyperparameters and considered values are provided in the Supplementary, Table 1.

3 PARTITION-BASED ENSEMBLE LEARNING

We consider a well-known supervised classification machine learning problem. The task is to find a partitioning of the dataset into K disjoint subsets $C_0, \dots C_{K-1}$ ($C_0 \cap C_1 \cap, \dots, \cap C_{K-1} = \emptyset$) such that the aggregated performance of models $M_0, \dots, M_{K-1}$ trained on these subsets is maximized.

3.1 Evaluation of fitness

The aggregated performance is the fitness measure in our EAs. To compute it, each model predicts class probabilities on the validation dataset (validation and train datasets do not overlap). Then, for each sample, these probabilities are averaged and the class with the maximum averaged probability is chosen as the final prediction. After obtaining predicted classes for all samples, the final fitness value is calculated as the standard accuracy metric. We use SVM as learners in constructed ensembles because of its good performance and computational efficiency (see Section 4.2).

We assume that fitness function evaluations are deterministic, i.e., the training of SVMs used in an ensemble is deterministic. Moreover, to count only truly different evaluations (without calculating solutions which define identical groups of samples twice i.e., 001122 encodes the same partitioning as 112200), all solutions along with their fitness values are stored in a normalized form. To do so, first, a solution is transformed to an actual partition: each subset k has N_k samples: $C_k = \{p_{i_0}, \dots, p_{i_{N_k-1}}\}$. Then, subsets C are sorted by the minimal index p of a sample belonging to a subset. A normalized solution form is then obtained by assigning values to variables according to the sorted order of subsets. Such solution normalization guarantees that different solutions defining the same dataset partition are evaluated and counted only once. When a new solution s is considered to be evaluated, first, it is normalized to s' and looked up in the collection of evaluated solutions. If s' has not been evaluated, then evaluation is performed, and the obtained fitness value is stored for s'. Note, since we consider the optimization function to be a black box, the normalized representation is only used to calculate fitness. The solution itself is left unchanged.

4 EXPERIMENTAL SETUP

4.1 Datasets

We consider five datasets from the OpenML-CC18 benchmark [7]. All datasets are supervised classification tasks and have at least 2000 samples. The selected datasets each have a different number of classes (between 2 and 11), and the number of features ranges from 16 to 57. All features are numeric. Specifications are provided in Table 1. From each dataset, 500 randomly selected samples are left out for the validation set, and the same number of randomly selected samples is left out for the test set. The remaining samples are shuffled and ℓ random samples are used for a considered optimization problem with ℓ variables (i.e., used for training).

Table 1. Datasets specification. All datasets are downloaded from the OpenML datasets collection.

Dataset	Samples	Features	Classes
segment	2310	16	7
spambase	4601	57	2
wall-robot-navigation	5456	24	4
kc1	2109	21	2
optdigits	5500	40	11

4.2 Ensemble training

We use standard regularized non-linear SVM with Radial Basis Function (RBF) kernel from the Scikit-Learn package. Before training, all features in the datasets are scaled to the unit interval. The regularization parameter of the SVM models is not tuned, but the default value of 1.0 is used. This choice is addressed in the Discussion.

4.3 Considered optimization algorithms

As simple baselines, we include in our experiments RS and LS. We also compare the performance of the proposed SA-P3-GOMEA to P3-GOMEA. Additionally, we consider three modern BO algorithms. COMBO [30] has shown state-of-the-art performance on several combinatorial optimization functions. Hyperopt [5] is an implementation of Tree Parzen Estimator (TPE). SMAC [25] is a BO algorithm specifically designed to handle categorical variables by using an RF surrogate model. Finally, we compare the achieved ensemble performance to training one SVM model on all samples.

4.4 Problem sizes and runtime budget

We consider various values of ℓ (i.e., the number of samples in the training dataset), and α (i.e., the number of subsets in a partition), specifically: $\ell \in \{100, 250, 500\}$ and $\alpha \in \{2, 5, 10\}$. On each dataset in each problem configuration (a combination of ℓ and α), all considered search algorithms perform 10 runs. Therefore, there are 50 runs by each algorithm for each problem instance.

As the global optima of the considered problems are not known, we compare the algorithms by the convergence trajectory of their averaged performance value. The averaged performance value after n real function evaluations is calculated as the elitist fitness function value after that many evaluations, averaged over all 50 runs.

In our main setup, the fitness function is deterministic and only unique dataset partitions are evaluated (as described in Section 3.1). P3-GOMEA, SA-P3-GOMEA, LS, and RS are run without a time limit until 5000 real function evaluations are performed. Additionally, we consider the case of noisy fitness evaluations where ensemble training is stochastic, i.e., re-evaluating the same partition results in a different fitness. In that case, all evaluations are counted.

We found that the considered BO algorithms, especially COMBO and SMAC, are much slower than P3-GOMEA and SA-P3-GOMEA.

We therefore set a time limit of 3 hours for BO algorithm runs for $\alpha = 2, 5$ and 5 hours for $\alpha = 10$. These values correspond to the observed upper bounds (rounded to hours) of SA-P3-GOMEA run time for corresponding problem instances. We found that even for the smallest considered problem ($\ell = 100, \alpha = 2$), COMBO could produce ℓ randomly solutions in the intialization phase, but only 3 additional solutions in the subsequent search phase after 4 hours of running. Hence, we do not include COMBO in the presented results as within the given time limit it works basically as RS.

4.5 Implementation details

Fitness functions for all conducted experiments are implemented in Python (version 3.7) with Scikit-Learn, and NumPy packages. All considered search algorithms use identical fitness functions. Code of COMBO[1], SMAC[2], and TPE[3] algorithms are downloaded from the corresponding repositories. P3-GOMEA, SA-P3-GOMEA, and LS algorithms are implemented in C++ with calls of Python fitness functions. All implementations are available at the repository of the first author[4]. Datasets are downloaded using the OpenML Python API. Experiments are conducted on a machine with Intel(R) Xeon(R) Silver 4110 CPU.

5 RESULTS

First, we analyze whether two hyperparameters, the relaxation hyperparameter η (described in Section 2.4.1) and the surrogate model type (described in Section 2.4.2), have a significant effect on performance. We test all surrogate models with $\eta = 0.99$ and $\eta = 0.999$. We do not consider lower η values, as they imply a more elaborate use of real fitness evaluations, leading to a performance close to non-surrogate P3-GOMEA. These results are provided in Figure 1. We conclude that 1) while both η values provide solid performance, $\eta = 0.999$ is slightly preferable and hence, we use it in further experiments; 2) SVR provides the best performance among the considered types of surrogate models. RF is a close runner-up, while MLP and Gradient Boosting models perform worse. We hypothesize that the better performance obtained with SVR is because SVR is less susceptible to overfitting and can generalize well from a limited number of training samples (i.e., extrapolate to unseen solutions). Moreover, it has fewer tunable hyperparameters than, for instance, Gradient Boosting models, and it is feasible to carefully tune them using grid search. Therefore, in further experiments, when we refer to SA-P3-GOMEA, SVR is always used as the surrogate model.

Next, we compare the results of SA-P3-GOMEA to other search algorithms (see Figure 2; Supplementary, Table 2). We observe that P3-GOMEA outperforms LS and RS in most cases. Moreover, the difference becomes larger as the number of variables increases. In general, SA-P3-GOMEA demonstrates solid performance on all problem instances. It always achieves the best accuracy within 500 evaluations, except for the smallest considered problem instance ($\ell = 100, \alpha = 2$) where after 2000 evaluations SMAC finds better solutions. We observe that SMAC does not scale well to larger problem instances as its computational overhead grows fast when both ℓ and α increase. In case of $\alpha = 10$ the problem apparently becomes too difficult within the given evaluations limit, as the achieved accuracy is below the baseline for $\ell = 100, 250$ and the performance of SA-P3-GOMEA is similar to P3-GOMEA. We suppose that in case of $\alpha = 10$ the surrogate modelling task is of too high dimensionality ($\ell\alpha$) and therefore, more solutions are needed to get accurate fitness function predictions. However, importantly, due to the adaptive nature of our new surrogate mechanism, even when it is not possible to accurately estimate fitness values, the performance of SA-P3-GOMEA does not become worse than P3-GOMEA.

From the practical perspective of the goal of ensembling, it is further important to note that a better validation accuracy is achieved than the baseline in most cases. Though the accuracy decreases as α increases, we believe it is possible to find better solutions if more function evaluations are allowed.

5.1 Runtime

On problem instances with $\alpha = 2, 5$ the runtime of SA-P3-GOMEA is below 2.5 hours. For $\alpha = 10$, runtime does not exceed 4.5 hours.

5.2 Statistical significance tests

We employ statistical hypothesis testing to compare the performance of SA-P3-GOMEA with P3-GOMEA and TPE. For each dataset and each value of ℓ (5 datasets, 3 values of ℓ) we do a pairwise statistical test to test a hypothesis that one algorithm performs better than another. Performance is assessed as the best achieved real fitness value during a run. We use the Wilcoxon pairwise test with Holm correction, resulting in a corrected significance level of 0.1. The p-values are presented in Table 2. For most problem instances, SA-P3-GOMEA outperforms P3-GOMEA and TPE.

5.3 Generalization study

In Machine Learning, a validation dataset is usually used for tuning hyperparameters, model selection, etc. In our setup, we search for the best ensemble configuration based on the accuracy score on the validation dataset. However, to evaluate the generalization ability of the obtained ensembles, we also look at the results on test datasets, which are held out during the search. These results are not related to the evaluation of the optimization potential of the considered algorithms, but are important to assess the practical value of the considered ensembling technique. These results are presented in Supplementary, Figure 1. We observe that convergence curves on test datasets are similar to the ones on validation datasets. Also, SA-P3-GOMEA demonstrates better performance in most cases.

6 DISCUSSION AND FUTURE WORK

The main goal of this paper was to introduce a novel efficient surrogate-assisted EA for solving combinatorial non-binary optimization problems with expensive function evaluations by integrating a surrogate model into a P3-like EA through a new adaptive mechanism with only one hyperparameter. Though we found the value of $\eta = 0.999$ for this hyperparameter to work well for the considered problem, it might need tuning for different expensive optimization problems.

Below, we identify and discuss interesting research topics for future work. Through additional experiments, we also considered

[1] https://github.com/QUVA-Lab/COMBO
[2] https://github.com/automl/SMAC3
[3] https://github.com/hyperopt/hyperopt
[4] https://github.com/ArkadiyD/SAGOMEA

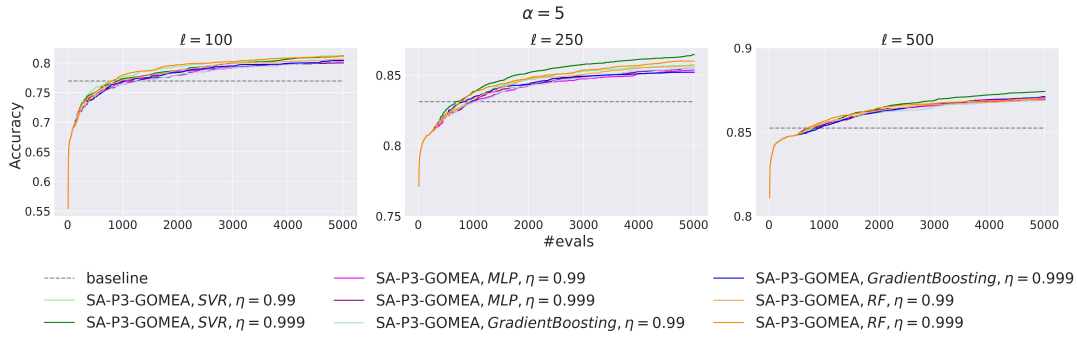

Figure 1. Convergence graphs of average performance for different SA-P3-GOMEA hyperparameters over five datasets and ten optimization runs for each dataset, for $\alpha = 5$ (partition into five subsets). Baseline denotes non-ensemble training using all samples in the training dataset, ℓ is the number of training samples, i.e., the number of variables in the optimization problem.

Table 2. Statistical significance testing of performance difference in pairs of best-performing algorithms. Reported p–values are obtained by pairwise Wilcoxon test and are uncorrected. Statistically significant results at $\alpha = 0.1$ with Holm correction ($m = 9$) are highlighted.

α	ℓ	**SA-P3-GOMEA** is better than **P3-GOMEA**	**SA-P3-GOMEA** is better than **TPE**
2	100	0.025	0.006
2	250	0.0	0.0
2	500	0.002	0.006
5	100	0.002	0.0
5	250	0.0	0.0
5	500	0.069	0.002
10	100	0.002	0.0
10	250	0.023	0.0
10	500	0.13	0.0

the case when fitness evaluations are noisy, i.e., when the same dataset partitioning can result in different ensemble accuracy scores because learners in ensembles are initialized with different random seeds. The results for $\alpha = 5$ are provided in the Supplementary, Figure 2. SA-P3-GOMEA performs better than P3-GOMEA for $\ell = 100, 250$ and not worse for $\ell = 500$. However, we observe that TPE performs significantly better for $\ell = 250, 500$. We note that, in contrast to TPE, neither P3-GOMEA nor SA-P3-GOMEA is specifically designed for solving noisy optimization problems. Potentially, specialized design choices aimed at solving noisy problems can improve their performance. However, the achieved accuracy values are much lower compared to the deterministic fitness function setup. We therefore conclude, that for practical usage deterministic ensemble training is preferable. Not only does it make the search problem easier for all considered algorithms, better solutions with fewer function evaluations can ultimately be found.

We chose SVM models as learners in the ensemble due to the solid tradeoff of SVM between computational efficiency and good performance. To achieve the best possible predictive accuracy, one may want to use modern Machine Learning models such as Gradient Boosting, or a mixture of models of different types with carefully tuned hyperparameters. Using the proposed approach to achieve state-of-the-art performance on a supervised machine learning problem is an interesting potential future use case.

We did not conduct experiments with more expensive function evaluations, e.g., using deep learning models in an ensemble. As we aimed to study and analyze the performance of the introduced surrogate-assisted EA in multiple setups, including deep learning models was not computationally feasible. However, validating the performance of SA-P3-GOMEA with such learners in an ensemble for specific applications is considered important future research.

Related to this, all considered types of surrogate models are general-purpose regression models and are not designed specifically for the dataset partition problem. We believe that it is possible to further improve the performance of SA-P3-GOMEA if specialized surrogate models, capable of explicitly modelling the task at hand, are designed. Developing such specialized models is however non-trivial and therefore considered a separate research topic.

7 CONCLUSION

We have introduced a novel surrogate-assisted Evolutionary Algorithm for expensive combinatorial optimization problems based on a state-of-the-art model-based EA and a novel adaptive surrogate fitness evaluation control mechanism. The considered model-based EA is a variant of the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) that uses a Parameterless Population Pyramid (P3) scheme. To the best of our knowledge, this is the first time a surrogate model is integrated into a P3 scheme. We have demonstrated that SA-P3-GOMEA achieves state-of-the-art performance on the problem of partitioning a dataset for the purpose of ensembling Machine Learning models. We experimented with different types of surrogate models for fitness values estimation and found that, among the models considered, SVR provides the best performance. The proposed SA-P3-GOMEA outperforms non-surrogate-assisted P3-GOMEA, local search, and various Bayesian Optimization algorithms on various ensemble learning problems with up to 500 variables of cardinality 10.

ACKNOWLEDGMENTS

This work is part of the research programme Commit2Data with project number 628.011.012, which is financed by the Dutch Research Council (NWO). We thank the Maurits en Anna de Kock Foundation for financing a high-performance computing system.

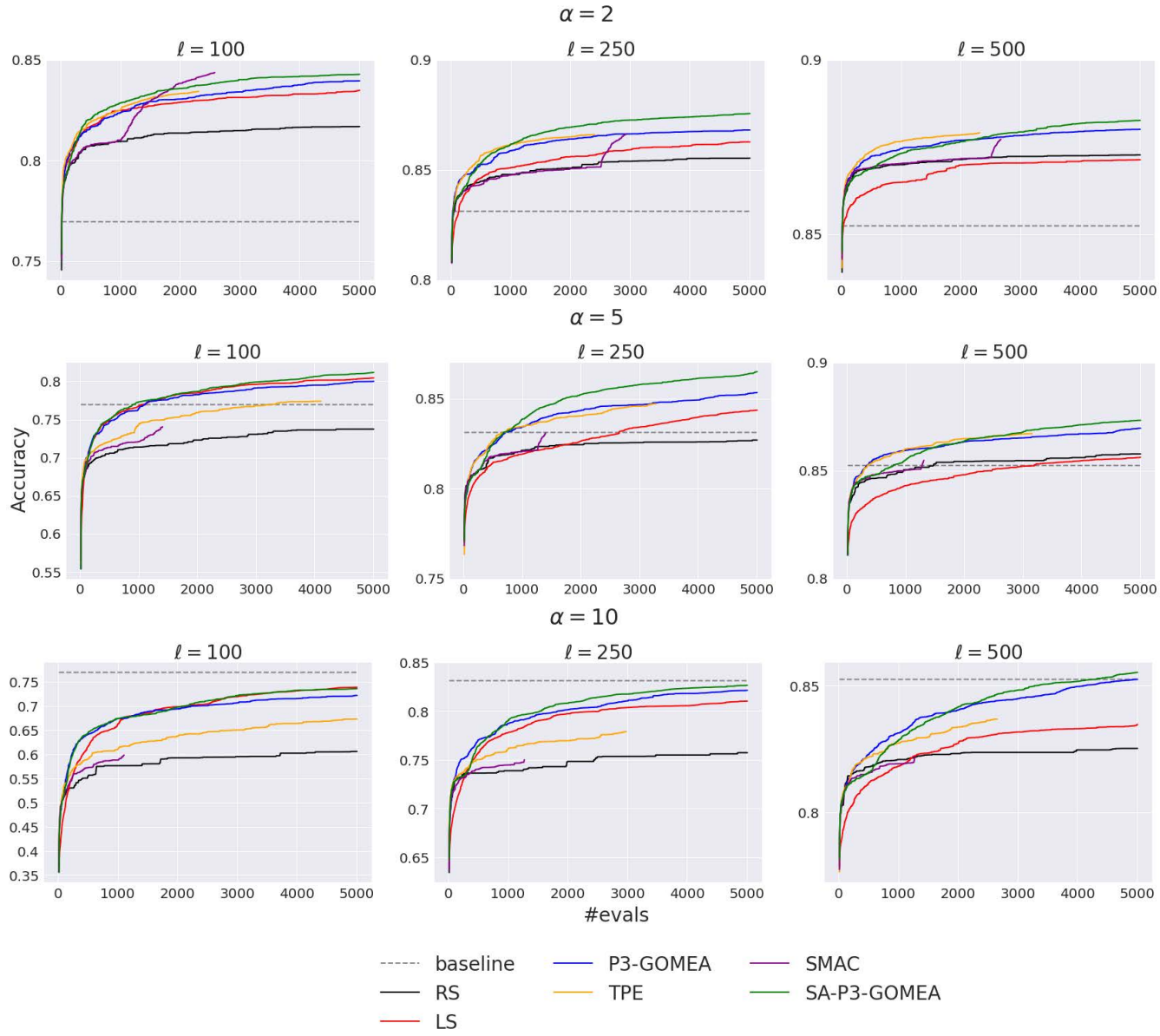

Figure 2. Convergence graphs of average performance for different values of α (the number of partition subsets). ℓ denotes the number of training samples, i.e., the number of variables in a corresponding optimization problem. Baseline denotes averaged validation accuracy when SVM is trained on all samples of training datasets of size ℓ. For TPE and SMAC, the values are presented for fewer than 5000 evaluations as these algorithms could not produce 5000 solutions within the time limit. Note that y-axis scales are different and depend on the accuracy range of the corresponding problem.

REFERENCES

[1] Ethem Alpaydin. 1997. Voting over multiple condensed nearest neighbors. In *Lazy learning*. Springer, 115–132.

[2] Ricardo Baptista and Matthias Poloczek. 2018. Bayesian optimization of combinatorial structures. *arXiv preprint arXiv:1806.08838* (2018).

[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*. 550–559.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems* 24 (2011), 2546–2554.

[5] James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*. PMLR, 115–123.

[6] Kalyan Shankar Bhattacharjee, Hemant Kumar Singh, Tapabrata Ray, and Juergen Branke. 2016. Multiple surrogate assisted multiobjective optimization using improved pre-selection. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 4328–4335.

[7] Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. 2017. OpenML benchmarking suites and the OpenML100. *arXiv preprint arXiv:1708.03731* (2017).

[8] Peter A. N. Bosman and Dirk Thierens. 2012. Linkage neighbors, optimal mixing and forced improvements in genetic algorithms. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. 585–592.

[9] Peter A. N. Bosman and Dirk Thierens. 2013. More concise and robust linkage learning by filtering and combining linkage hierarchies. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. 359–366.

[10] Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24, 2 (1996), 123–140.

[11] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.

[12] Nitesh V. Chawla, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. 2004. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research* 5 (2004), 421–451.

[13] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.

[14] Zhuo Chen, Fu Jiang, Yijun Cheng, Xin Gu, Weirong Liu, and Jun Peng. 2018. XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud. In *2018 IEEE international conference on big data and smart computing (bigcomp)*. IEEE, 251–256.

[15] Willem den Besten, Dirk Thierens, and Peter A. N. Bosman. 2016. The Multiple Insertion Pyramid: A Fast Parameter-Less Population Scheme. In *International Conference on Parallel Problem Solving from Nature*. Springer, 48–58.

[16] Tom Den Ottelander, Arkadiy Dushatskiy, Marco Virgolin, and Peter A. N. Bosman. 2020. Local Search is a Remarkably Strong Baseline for Neural Architecture Search. *arXiv preprint arXiv:2004.08996* (2020).

[17] Alan Díaz-Manríquez, Gregorio Toscano, Jose Hugo Barron-Zambrano, and Edgar Tello-Leal. 2016. A review of surrogate assisted multiobjective evolutionary algorithms. *Computational Intelligence and Neuroscience* 2016 (2016).

[18] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363* (2018).

[19] Arkadiy Dushatskiy, Adriënne M. Mendrik, Tanja Alderliesten, and Peter A. N. Bosman. 2019. Convolutional neural network surrogate-assisted GOMEA. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 753–761.

[20] Arkadiy Dushatskiy, Adriënne M. Mendrik, Peter A. N. Bosman, and Tanja Alderliesten. 2020. Observer variation-aware medical image segmentation by combining deep learning and surrogate-assisted genetic algorithms. In *Medical Imaging 2020: Image Processing*, Vol. 11313. International Society for Optics and Photonics, 113131B.

[21] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. 2019. Neural architecture search: A survey. *J. Machine Learning Research* 20, 55 (2019), 1–21.

[22] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. 2020. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing* 380 (2020), 20–35.

[23] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. 2019. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Brian W. Goldman and William F. Punch. 2015. Fast and efficient black box optimization using the parameter-less population pyramid. *Evolutionary Computation* 23, 3 (2015), 451–479.

[25] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*. Springer, 507–523.

[26] Jack P. C. Kleijnen, Wim van Beers, and Inneke van Nieuwenhuyse. 2010. Constrained optimization in expensive simulation: Novel approach. *European Journal of Operational Research* 202, 1 (2010), 164–174.

[27] Slawomir Koziel and Leifur Leifsson. 2015. Efficient knowledge-based optimization of expensive computational models using adaptive response correction. *Journal of Computational Science* 11 (2015), 1–11.

[28] Ngoc Hoang Luong, Marinus O. W. Grond, Han La Poutré, and Peter A. N. Bosman. 2015. Scalable and practical multi-objective distribution network expansion planning. In *2015 IEEE Power & Energy Society General Meeting*. IEEE, 1–5.

[29] Zahra Mirikharaji, Kumar Abhishek, Saeed Izadi, and Ghassan Hamarneh. 2020. D-LEMA: Deep Learning Ensembles from Multiple Annotations–Application to Skin Lesion Segmentation. *arXiv preprint arXiv:2012.07206* (2020).

[30] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. 2019. Combinatorial Bayesian optimization using the graph Cartesian product. In *Advances in Neural Information Processing Systems*. 2914–2924.

[31] Kashif Rashid, William J Bailey, Benoit Couet, David Wilkinson, et al. 2013. An efficient procedure for expensive reservoir-simulation optimization under uncertainty. *SPE Economics & Management* 5, 04 (2013), 21–33.

[32] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4780–4789.

[33] Dirk Thierens and Peter A. N. Bosman. 2011. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 617–624.

[34] Colin White, Sam Nolen, and Yash Savani. 2020. Local Search is State of the Art for NAS Benchmarks. *arXiv preprint arXiv:2005.02960* (2020).

[35] Jiancheng Zhong, Yusui Sun, Wei Peng, Minzhu Xie, Jiahong Yang, and Xiwei Tang. 2018. XGBFEMF: An XGBoost-based framework for essential protein prediction. *IEEE Transactions on NanoBioscience* 17, 3 (2018), 243–250.