A DIFFERENTIAL EVOLUTION APPROACH TO FEATURE SELECTION IN GENOMIC PREDICTION

By

Ian Whalen

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Master of Science

2018

ProQuest Number: 13419545

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 13419545

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

ABSTRACT

A DIFFERENTIAL EVOLUTION APPROACH TO FEATURE SELECTION IN GENOMIC PREDICTION

By

Ian Whalen

The use of genetic markers has become widespread for prediction of genetic merit in agricultural applications and is a beginning to show promise for estimating propensity to disease in human medicine. This process is known as genomic prediction and attempts to model the mapping between an organism's genotype and phenotype. In practice, this process presents a challenging problem. Sequencing and recording phenotypic traits are often expensive and time consuming. This leads to datasets often having many more features than samples. Common models for genomic prediction often fall victim to overfitting due to the curse of dimensionality. In this domain, only a fraction of the markers that are present significantly affect a particular trait. Models that fit to non-informative markers are in effect fitting to statistical noise, leading to a decrease in predictive performance. Therefore, feature selection is desirable to remove markers that do not appear to have a significant effect on the trait being predicted. The method presented here uses differential evolution based search for feature selection. This study will characterize differential evolution's efficacy in feature selection for genomic prediction and present several extensions to the base search algorithm in an attempt to apply domain knowledge to guide the search toward better solutions.

ACKNOWLEDGEMENTS

I would first like to thank my referee committee: Cedric Gondro, PhD, Charles Ofria, PhD, and my advisor Wolfgang Banzhaf, Dr.rer.nat. You have each contributed to my academic journey through unique and invaluable ways. This thesis topic would not have been possible without Dr. Gondro's collaboration and genomics expertise. Dr. Ofria was first to spark my interest in evolutionary computation and the process of conducting research. I will always think back fondly of my time at the Devolab. Finally, Dr. Banzhaf has shown nothing but unwavering support for me during my whirlwind tour of graduate school. I cannot express my gratitude enough for his guidance and encouragement.

More generally, I would like to thank the BEACON Center for the Study of Evolution in Action. The multidisciplinary spirit inspired by BEACON has contributed volumes to my personal and professional life. BEACON funded this work under NSF Cooperative Agreement No. DBI-0939454. Thanks also to Michigan State University's Institute for Cyber Enabled Research. Their high performance computing center—and its timely software update—was a cornerstone for the completion of this thesis. Finally, thanks to my family and friends. I hope to see more of you now that this document is complete.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Using DNA markers for prediction of genetic merit has become widespread in plant and animal breeding and is gaining momentum as a prognostic tool for susceptibility to disease in human medicine. Meuwissen et al. [1] introduced the idea of using a very large number of markers to predict phenotypes. This process is known as genomic prediction and tasks a system with estimating the effects that thousands of markers have on a trait. For agricultural applications, these effect estimations can be used to predict phenotypes or breeding values for new individuals that do not have trait information but do have marker information. Over the past ten years, genomic prediction has been widely adopted for genomic selection [2] in agriculture [3, 4, 5, 6] and in human studies [7, 8]. Hayes, Bowman, Chamberlain, and Goddard [9] emphasize its value, touting genomic selection as the most significant advancement for the dairy industry in the last two decades with the potential to double the rate of genetic gain in particular yield indices. Hayes, Lewin, and Goddard further emphasize its utility in [10] by pointing out valuable future objectives of genomic prediction like reducing methane emission by ruminants[1] to potentially mitigate the contribution the livestock industry has to climate change.

Although conceptually straightforward, genomic prediction is a very challenging problem. Genotyping and trait recording are costly and time demanding processes. The result is that most genomic datasets will have hundreds of thousands or even millions of markers for which effects need to be simultaneously estimated from usually only a few thousand phenotyped individuals. This means that the datasets are underdetermined (also known as the $p \gg n$ problem in bioinformatics). Models trained on such data suffer from overfitting due to the curse of dimensionality. In effect, genomic prediction can be treated as a high dimensionality, sparse data problem and, consequently, suffers from the same issues as other problems of the same type. Statistical prediction models are

[1] Animals that digest through the rumination process which ferments normally indigestible plant matter into useful byproducts [11].

suboptimal in this case since the accuracy marker effect estimates rapidly decays as the number of features to be estimated increases. The accuracy of prediction is also conditional on the genetic architecture of the traits – the interplay between genotypes and phenotypes is complex and varies widely from trait to trait; e.g. highly heritable traits regulated by a few genes of large effect are easier to predict than traits regulated by many genes with small effects and with a low heritability [12]. Moreover, there are still various other factors that will also influence the accuracy of prediction such as marker density (if the data is not at full sequence resolution), measures of linkage disequilibrium and family relationships [13, 14, 15], population stratification [16], sample size, reliability of phenotypes [17], and, of course, the methodology used to estimate marker effects [14].

Since they are characteristically underdetermined, genomic datasets are prime candidates for feature selection techniques. However, popular methods for genomic feature selection are often statistically-based; e.g. genome wide association studies (GWAS) which aim to identify, in the case of sequence data, the causal variants of a given trait or, when single-nucleotide polymorphism (SNP) arrays are used, the markers that are in high linkage disequilibrium with the causal variants [18, 19]. Such approaches are limited to local searches of the feature space since they are conditioned on the supporting statistical evidence. Without feature selection, all markers must be simultaneously estimated for an effect value without knowledge of their possible influence on a trait.

Even though quantitative traits are largely polygenic, with hundreds or thousands of variants influencing a trait, it still stands to reason that not every single genetic variant across the genome will have a real effect on a trait. This suggests that current methods lead to suboptimal accuracy in genomic prediction due to background noise introduced by the large number of uninformative effect estimations included in a model. Under this rationale, it stands to reason that better models are attainable by using subsets of markers. Hence, genomic prediction should be treated as a feature selection problem that is amenable to non-statistical methods since they are potentially better at performing global searches of the feature space. The following study will discuss a non-statistical approach for genomic prediction through the use of an evolutionary computation (EC) technique called differential evolution (DE) [20] and compare its performance to mainstream

genomic prediction methods.

CHAPTER 2

BACKGROUND

2.1 Microarray Data

Microarray data is a general term used to describe data that represents something about organism's genome. The use of microarray data has been the bedrock of many in human medicine and agricultural. There are many types of microarrays such as protein, peptide, antibody, and DNA. One of particular interest in this study is called a single-nucleotide polymorphism (SNPs) microarray. As the name suggests, SNPs describe the phenomenon of a nucleotide in DNA switching to another base, e.g. a "C" changing to "A" in some percent of a population. This results in ternary datasets, where 0 and 2 represent to the homozygous genotypes (AA, BB) and 1 represents to the heterozygous genotype (AB). SNP markers are one of the most commonly used ways to measure genetic variation [21]. Forgoing the details of SNP detection, it suffices to say that SNPs allow for a more compact representation of an organism's genotype that highlights parts of the genome that vary. SNPs are numerous in any organism's genome. For reference, the human genome has upwards of 11 million SNPs [22], and have been estimated to occur in about every 500 to 1000 base pairs [21].

2.2 Genome Wide Association Studies

Genome wide association studies (GWAS) seek identify which markers significantly affect a particular trait. In human trials, this comes in the form of identifying genetic risk factors for a particular disease [18, 23]. For agriculture and livestock applications, GWAS can discover genetic factors that affect economically desirable traits like crop yield per hectare [6] or cattle muscle marbling [24]. More concretely, a GWAS attempts to find what are called the quantitative trait loci (QTL). A quantitative trait is a phenotype that is the cumulative effect of many markers and the environment [21].

A simple method for GWAS is to use a trait and single SNP regression tests to determine a p-value for each marker [25]. Such a test lacks the ability to estimate or consider the interactions between traits since each SNP is regressed independently. Such tests can be carried out by calculating common statistics like ANOVA or an F-regression with the standard null hypothesis being that all SNPs have an effect of 0 on the trait in question [25]. Normal significance testing would allow for a threshold of, for example, 0.05 to reject the null hypothesis of a zero effect. However, each test is an independent trail. Many such trials increase the probability of a false positive when rejecting the null hypothesis. Therefore, some adjustment must be made in order to sort out which SNPs are actually significant. This is done using the Bonferroni adjustment [26]. The adjustment is a simple ratio of the chosen significance threshold and the number of SNPs used in the statistical trials, namely,

$$\alpha_{GWAS} = \frac{\alpha}{n_{SNPs}}. \tag{2.1}$$

This results in a dramatic change in the significance threshold, for example a human trial using 11 million SNPs would have $\alpha_{GWAS} \approx 5x10^{-9}$.

2.3 Genomic Prediction

Genomic prediction is the process of a system predicting an organism's phenotype given information about its genotype—usually in the form of SNPs. Meuwissen, Hayes, and Goddard first introduced the idea of using a large number of genotype markers to predict a trait [1]. In contrast to GWAS, genomic prediction usually attempts to use all of a population's markers at once to predict a trait, rather than analyze the significance any one marker has on a trait. Genomic prediction has been shown to outperform simple GWAS in predicting complex traits and is hypothesized to value the relationships between a population of animals more than specific markers [27]. Genomic prediction has shown recent success across both agricultural application and human trials.

In agriculture, genomic prediction is mainly used for a process called genomic selection. Genomic selection seeks to automate the selection process using genetic markers to predict desirable traits—i.e., estimated breeding values (EBVs)—of a population of plants or animals [2]. His-

torically, the selection process was accomplished with selective breeding. This practice is well documented by classical naturalists like Darwin [28]. In the 20^{th} century, selective breeding was formalized by Henderson to use pedigree information to predict EBVs [29]. The current state of the art replaces simple pedigree descriptions of relatedness with genomic relatedness described by what is known as the genomic relationship matrix [21]. See Sections 3.1.1.2 and 3.1.1.3 for specifics on how this prediction is carried out.

Genomic prediction for human trials is relatively new and is used for computing genetic risk values for particular complex diseases. Diseases like coronary heart disease [30], diabetes [31], cancer [32], schizophrenia [33], and celiac disease [34] are affected by many markers. Predicting such diseases is a classification task that can use methods like logistic regression [30] to discriminate by outputting the probability a certain individual has or will develop a disease. Risk prediction can be seen as a classical machine learning problem, where predictive accuracy metrics like area under ROC curve are maximized [34]. Recently, agricultural techniques discussed above have been applied to predicting celiac disease [34]. Human research has been mostly confined to the study of disease through genetic risk, although some work has been done to predict quantitative traits like height and heel bone density [35].

2.3.1 Linear Methods

Linear methods follow the standard assumptions found in any linear regression model. More concretely, such methods approximate an infinitesimal model that assumes all markers contribute some nonzero effect to genetic variance. Furthermore, it is assumed that all marker effects are normally distributed. The following discussion presents the functionally equivalent [36] methods: SNPBLUP [37] and GBLUP [38, 39]. These methods are well studied and have been applied to many tasks [40, 41, 42].

SNP best linear unbiased predictor (SNPBLUP) is a simple method that applies classical ridge regression [43] to marker assisted selection [37]. The original motivation for SNPBLUP was to askew the necessity for marker subset selection and allow the optimization process to determine

important features. Genomic best linear unbiased predictor (GBLUP) takes a similar approach, but rather than using the full data matrix, employs a genomic relationship matrix to estimate effect values. See Sections 3.1.1.2 and 3.1.1.3 for specifics on how these processes are carried out on SNP data.

It is important to point out that these methods share a common flaw. As mentioned above, they both assume that each marker contributes a nonzero effect value. This leads to markers that do not affect a trait being assigned an effect value. Though the value may be small, two problems arise when fitting to uninformative markers. First, the effect value assigned is essentially fit to statistical noise and has no meaning. Second, genomic datasets often have tens or hundreds of thousands of markers. This causes these uninformative marker effect values to compound and reduce model performance. Therefore, feature selection is desirable to attempt to remove markers that appear to have effect on trait before building a model.

2.3.2 Non-Linear Methods

Mainstream non-linear methods for genomic prediction include BayesA, BayesB [1], BayesC [44], Bayesian Lasso [45], and BayesR [46]. These methods—among others—are known as the Bayesian alphabet [44]. Such techniques mainly differ in their assumptions about what prior distributions the marker effects should follow. Even though a large proportion of markers might be allocated to a distribution with very small to zero effects, these methods will still assign a nonzero posterior density to most variants. Hence, the number of markers to be used for prediction is still large. Like linear methods, this causes Bayesian methods to struggle to discriminate between markers with and without an actual effect. Furthermore, these methods often become computationally intractable at very high dimensionalities due to iterative sampling of each effect distribution. The following presents the motivation of the original BayesA and B [1] as well as background on the more state of the art BayesR [46].

The original motivation of applying Bayesian regression to genomic prediction in [1] was to allow for estimation to reliably assign a zero effect value to uninformative markers [46]. In

summary, BayesA and B construct a more reliable posterior distribution for marker effects and iteratively sample from the distribution with Gibbs sampling and assign SNP effect values with a Metropolis-Hastings algorithm [1]. Erbe et al. critique this method in [46], pointing out that BayesA and B are more computationally taxing than necessary and the assigned effect values too closely follow the assumed prior distribution. Their proposed method, BayesR, has SNP effects follow a simple mixture of Gaussians and showed an increase in predictive accuracy over BayesA [46]. The computational efficiency of this method was later improved through an expectation maximization approach rather than Gibbs sampling [47].

Recently, non-linear machine learning approaches have been applied to genomic prediction as well. Li et al. [48] show promising results using random forest [49, 50], gradient boosting machines [51], and extreme gradient boosting [52] to effectively identify subsets of SNPs and predict traits in cattle. These methods identify SNPs through intrinsic methods discussed more thoroughly in Section 2.4.2. Grinberg et al. also show similar results with random forests, finding that the technique outperforms GBLUP in some cases on an agricultural application [53]. Machine learning applications that compare with traditional methods (e.g. [53]) like SNPBLUP and state of the art methods such as BayesR often present the most compelling results since they have a frame of reference to the community at large. A future unification of the principles in machine learning and quantitative genomics would likely further model performance. To conclude non-linear methods for genomic prediction, background on deep learning methods is presented.

Deep learning for genomic prediction is in very early stages and is mostly valued for its ability to extract marker embeddings. This is a form of dimensionality reduction that learns some mapping between the genotype space and a lower dimensional space, often through convolutional neural networks (CNNs). Zhou and Troyanskaya present one of the earliest attempts to apply deep learning to marker data with the deep learning-based sequence analyzer (DeepSEA) [54]. DeepSEA uses a CNN to extract features and showed improvement over traditional GWAS when identifying functional SNPs and predicting human traits. However, the dataset used in [54] was small in comparison to standard deep learning datasets. Since deep learning is well-known for

needing massive amounts of data, it is likely performance was affected. More recently, Bellot, de los Campos, and Pérez-Enciso [55] applied deep learning to the well-known UK Biobank (`www.ukbiobank.ac.uk`) dataset. Their specific subset of the original data contained 102,221 samples and 567,867 usable SNPs, a dramatic increase in sample size over typical genomic prediction datasets. Genotypes were fed into a CNN and showed a competitive performance when compared to traditional genomic prediction methods [55]. Finally, methods that treat the genome as a sequence (like text or audio) rather than a fixed set of features have also shown some success. For example, Quang and Xie showed that a CNN paired with a special type of network called a recurrent neural network showed a 50% increase in performance over related models in predicting the function of non-coding regions of DNA [56]. However, models that treat markers as a sequence have not yet been applied to genomic prediction.

2.3.3 Heritability

Discussion on genomic prediction is not complete without defining the concept of heritability. Specifically, this definition focuses on "narrow sense" heritability, which is the total phenotypic variance that is captured by only additive effects in the genotype [21], i.e. the effect of certain markers being present is cumulative—rather than depending on particular combinations of markers (epistatic effects). To define this heritability two other measures of variance must be discussed first. First, σ_A^2, is the contribution to genetic variance from individual alleles. Second, is $\sigma_P^2 = \sigma_G^2 + \sigma_E^2$, where σ_G^2 is the total variance from the genome and σ_E^2 is the variance from the environment. Then, heritability is defined by

$$h^2 = \sigma_A^2 / \sigma_P^2, \tag{2.2}$$

as shown in [21]. Heritability is a fairly unique characteristic to genomic prediction, as it can be used as a proxy for maximum performance. The value h (i.e., $\sqrt{h^2}$) is effectively the theoretical limit on predictive performance for any genomic prediction model. This will be revisited in later discussion, see Section 3.6. In practice, heritability can only be estimated. The true variance of the

genotype and environment are hard—if not impossible—to directly study. Heritability estimation is done through analyzing the behavior of a trait through familial lines using a restricted maximum likelihood technique [57, 58].

2.4 Microarray Feature Selection

Feature selection is a subset of a broad group of techniques known as dimensionality reduction. Feature selection seeks to select a single "best" subset of a d dimensional dataset from 2^d possible subsets. This is in contrast to feature extraction (or transformation) that, in general, attempts to apply a function to the set of all features to reduce the overall dimension. Feature selection can be separated into three distinct classes: filter, embedded, and wrapper methods [59]. Recently, non-statistical wrapper and embedded methods have gained popularity as they can often perform global feature space searches and more easily take into account possible epistatic interactions, i.e. interactions that are conditional on pairs, triplets, or larger tuples of markers all being a specific value. This section will focus specifically on methods for feature selection in microarray data. Some background included here is tangential to genomic prediction, but is included to give an overview of methods for feature selection in bioinformatics.

2.4.1 Filter Methods

Filter methods are the most straightforward of the feature selection methods. These methods use statistical properties of data and function independently of learning, which makes them useful as a preprocessing step to remove noise. A feature is selected based on some statistical relationship it has with the output being predicted—like correlation [60]—or its relationship with other features—like redundancy [61]. Filter methods work by suppressing the least interesting variables, leaving the more promising variables to be used to train a predictive model. Filter methods tend to be computationally efficient and are robust to overfitting, making them a popular choice for genomic prediction tasks involving classification [62, 63, 64].

The standard GWAS discussed in Section 2.2 can be used as a filtering method. For SNP data,

the standard single SNP regression (SSR) can be carried out to obtain p-values for each marker. Then a model can be trained using the markers that are deemed significant. For certain traits, this is likely to perform well. However, for most quantitative traits studied in genetic risk and genomic selection problems, this is often suboptimal. Most critically, a SSR technique does not take into account population structure, which is imperative to accurate prediction [65]. More subtly, an SSR also does not take into account interactions between markers. Though some attempts have been made at using mutual information filtering for microarray data [66]. Extensions of these ideas to regression—e.g. RReliefF [67]—often do not generalize well and fall short in very high dimensional problems with weak signals [68, 69] that are characteristic in genomic prediction.

2.4.2 Embedded Methods

Embedded methods—as their name suggests—are built into an existing prediction model. As the model is optimized, a feature subset is naturally extracted through various properties of the learning model. Such models include Lasso regression [70, 71], random forest (RF) [49, 50], gradient boosting machine [51], and support vector machine (SVM) [72]. The following discussion will focus on the use of RF and SVM for feature selection in microarray datasets.

Breiman explains the process of extracting a feature importance ordering using the original implementation of RF, in [50]. The technique hinges on *bagging* which is the process of sampling the dataset with replacement to train the decision trees in a random forest. To compute the importance measure of each feature, its order is randomly permuted, and the samples are bagged again to compute "out-of-bag" error for each decision tree in the forest. This error is then averaged across all trees. Features with the highest mean error are considered most important because model performance decreased the most when those particular features were randomly permuted. Since this error is computed anyway during training a RF, some implementations report these feature importance orderings without requiring any extra work, e.g. [73]. Some improvements to the training process have been proposed to help select markers in disease prediction tasks [74] and work has been done to optimize RF for higher dimensional datasets [75]. Li et al. [48] demonstrated

the performance of RF on a cattle genomic prediction task which, in certain cases, outperformed common genomic prediction methods.

The SVM was first introduced by Boser, Guyon, and Vapnik in [72]. Its benefits to high dimensional problems are two-fold. First, through the dual of the SVM objective function, dimensionality has little bearing on how quickly the model can be optimized. Second, the SVM is inherently self-regularizing and thus the largest weights can be observed after training to extract a feature subset—similar to the non-zero weights in Lasso regression. The SVM can be repeatedly trained and non-important features removed until accuracy does not improve in a process called SVM recursive feature elimination [76]. This technique has shown success in eliminating gene redundancy in cancer classification sets [77, 78]. However, it does not consider interactions between features [77, 76] and, again, no significant work has been done in regression tasks.

2.4.3 Wrapper Methods

A wrapper method iteratively updates its feature subset over time, preferring subsets that perform better according to some measure—usually performance metrics of a predictive model trained with the selected subset. The simplest wrapper method is an exhaustive search of the entire feature space. For a given predictive model, this is guaranteed to select the optimal subset from the 2^d possible subsets of a d dimensional dataset. However, this is obviously an impossible task due to computational limits as d grows to any size typical of microarray datasets. Therefore, approximate methods that obtain a suboptimal feature subset are necessary in practice. The two main wrapper methods that scale well in practice are sequential methods and population-based methods [76, 79].

Sequential methods are a deterministic way of selecting feature subsets. Sequential methods either select features in a forward manner—starting from an empty set and adding features—or a backward manner—starting from evaluating all features at once and removing features. For simplicity, consider sequential forward selection (SFS) [80]. At every update of SFS, each feature not included in the current subset is added, the objective value of the subset is calculated, then the feature is removed. This process repeats until each feature is tested. Whichever feature increased

the objective function the most is added permanently and the process repeats. In the end, the objective function will be evaluated $d * (d + 1)/2$ times for a d dimensional dataset. Furthermore, permanent addition of features in SFS leads to an issue known as "nesting" that can be addressed with a "floating" method as described by Pudil, Novovičová, and Kitter in [81] that allows for the removal of past features. Sequential forward selection is well-known in cancer applications involving microarray data [76] and is often used along with some cross-validation scheme like k-fold cross-validation to reduce selection bias in the validation set [82]. A common predictive model in many wrapper methods is SVM which is sometimes combined with the recursive scheme described in Section 2.4.2 to accelerate feature selection [77, 83].

Population-based methods include evolutionary computation (EC) approaches and particle swarm optimization (PSO) [84]. Evolutionary computation includes a diverse catalog of methods like genetic algorithms (GAs) [85], genetic programming [86], evolutionary strategies [87], differential evolution (DE) [20], and ant colony optimization [88]. See Section 2.5 for more on DE and its use in feature selection. These techniques are biologically influenced heuristic based search techniques that are valued for their abilities to search both globally and locally on difficult problems. Furthermore, EC and PSO have the luxury of being able to use almost any objective function—non-differentiable or otherwise—due to their gradient-free nature. There are consistent themes across both of these paradigms. A group of solutions to a problem, sometimes called a *population* of *individuals*, each have a *fitness* assigned to them based on a given objective function. Solutions then somehow combine—or share information—to guide the search toward the global optimum.

Evolutionary computation has been used to successfully select feature subsets on gene expression microarray data. Luque-Baena et al. [89] showed that a simple GA could outperform the state of the art on a cancer pathway identification and classification task. Furthermore, it was shown in [90] that a GA outperforms simple sequential feature selection methods. Feature subsets discovered in [90] were also deemed more biologically relevant, potentially furthering the understanding about the traits being predicted. The SVM makes an appearance in successful EC approaches as well.

Perez and Marwala present a hybrid approach that combines an SVM, GA, and simulated annealing for feature selection in cancer datasets [91].

Feature selection with PSO has also shown success in microarray applications. Tang, Suganthan, and Yao [92] showed a PSO approached using an SVM for selecting features outperformed sequential selection schemes as well as the recursive SVM scheme used in [77]. Li, Wu, and Tan combined PSO and a traditional GA in [93] for a cancer classification task. Their results outperformed almost all methods compared against, including the use of PSO and a GA separately. This speaks to an important feature of EC and PSO-like approaches: they are often easily combined with other approaches and domain knowledge of a particular problem to improve performance.

2.5 Differential Evolution

Differential evolution was originally proposed by Storn and Price as a greedier alternative to stochastic optimization methods of the time [20]. Moreover, they proposed four criteria for a practical optimization method: (1) the ability to handle non-differentiable objective functions, (2) ease of parallelization, (3) few hyperparameters, and (4) consistent convergence. The authors go on to demonstrate that DE satisfies these properties and show its efficacy compared to algorithms like GAs when optimizing an array of test functions, e.g. Rastrigin's function [94, 95]. DE is well-known for its simplicity—requiring only 3 hyperparameters: mutation factor (F), crossover rate (C_r), and population size (N_p)—and fast convergence properties [96].

Advances in DE have focused on alleviating the choice of hyperparameter and crossover scheme, leaving population size as the sole hyperparameter. Three well-known algorithms that remove these choices are JADE [97], SaDE [98], and MDE_pBX [99]. These methods hinge on sampling a mutation rate and/or crossover rate from a distribution for each individual in the population and creating a new individual with the sampled rates. The distributions are then updated based on some formula involving the number of "successful" offspring, i.e. newly created solutions that enter the next population. See Section 3.2 for more detail. These methods are convenient and have shown success on the usual battery of test problems, however some recent criticism has been made of

their efficacy in practice. Al-Dabbagh et al. [100] point out that some adaptive strategies have begun to venture into the realm of ad hoc solutions that are sometimes problem dependent and often lack analysis into the effect they have on sensitive hyperparameters. Self-adaptive methods leave population size to be chosen by the practitioner. This choice has been a major DE research area over the past decade and different algorithms behave quite differently with varying population sizes [96]. The following is a brief review of common real valued optimization tasks DE has been applied to and the applications of DE in feature selection. See Section 3.1 for a formal description of the DE algorithm.

2.5.1 Real Valued Optimization

As mentioned, DE is prized for its convergence properties in real valued optimization problems. Though the no free lunch theorems [101] lay out the fact that DE cannot be the best optimization algorithm in all problems, it has shown success across many applications. The first "application" that new DE algorithms are often applied to are the Congress on Evolutionary Computation (CEC) benchmarks titled CEC2005 [102] and CEC2011 [103]. These benchmarks were established to enforce consistency and create a common ground for comparison across a quickly expanding number of stochastic optimization algorithms. CEC2005 is a catalog of unimodal, multimodal, and composition functions that are difficult to optimize. CEC2011 is a collection of real-world optimization tasks gathered from various studies. DE has shown success outside of these benchmarks as well in tasks like power grid management [104], clustering [105], and various microarray applications [106, 107, 108].

2.5.2 Feature Selection

In order to do feature subset selection, a real valued vector must be somehow converted into a list of integers corresponding to indices of features in a data matrix. This is done using a technique known as random keys. First introduced by Bean, random keys was originally a technique to adapt real valued GAs to be a general, robust encoding and decoding scheme for combinatorial

problems [109]. Random keys can be used for problems like as feature subset selection, the traveling salesman problem [110], and scheduling. See Section 3.1.1.1 for a formal description. In summary, the random key encoding allows for any standard genetic operations like crossover and mutation to be carried out without violating the constraints defined in a problem.

Feature selection using DE is relatively new, with first successes being shown in 2008 [111]. In that work, DE was shown to outperform other wrapper methods like PSO and GAs in an electroencephalogram classification task. The method presented here is the technique in [107, 108]. Both works deal with using DE to do feature selection in cattle applications. Esquivelzeta-Rabell et al. show in [108] that DE feature selection performs well in selecting small, functional SNP panels when compared to random search in a breed prediction task. Al-mamun et al. [107] further show that a DE approach can outperform both BLUP and Bayesian Lasso on a simulated dataset task.

CHAPTER 3

METHODS

3.1 Differential Evolution

Differential evolution is a real-valued optimization technique originally introduced by Storn and Price [20]. The strategy for DE introduced here is the original implementation. See Algorithm 1 for an overview which is abstracted into four main parts: evaluation, mutation, crossover, and selection. These operations are carried out on a population of possible solutions to a problem. As is true with all evolutionary algorithms, the population is the foundation of DE. It is comprised of N_p candidate solutions, all of which are vectors in $\mathbb{R}^d$, where d is the dimensionality of the given problem. Each solution, $\mathbf{X}_i = \left[x_{1,i}, \ldots, x_{d,i}\right]$ has an associated *fitness*, which is assigned by an objective function. Before a DE search begins, each vector is randomly initialized according to a uniform random distribution, so that $0 \leq x_{j,i} < 1, \forall j$. Traditionally, in an evolutionary algorithm, $\mathbf{X}_i$ would be referred to as a *genome*. However, given the application presented in this paper, *solution* or *solution vector* will be preferred to avoid confusion with biological genomes.

As mentioned, an abstraction of DE is favored in Algorithm 1 as there are many possible way to perform each of the operations. The rest of this section is organized into the descriptions of operations applied to the population: Section 3.1.1 covers assigning each solution vector a fitness, Sections 3.1.2 and 3.1.3 cover mutation and crossover, and Section 3.1.4 presents selection. Each repetition of these stages is called a *generation*. Most often, this process is repeated until some fixed number of generations, g.

3.1.1 Evaluation

Each vector in the population has an associated fitness that is assigned by an objective function. Here, to obtain a fitness value we first subset the genotype matrix using a solution vector from the population. See Section 3.1.1.1 for discussion on obtaining this subset. Next, we compute a

Algorithm 1 Differential Evolution

Input: Population size N_p, dimensionality d, generations g
Output: Solution P_{best}

$P = \{\mathbf{X}_i | \mathbf{X}_i \in [0, 1)^d, 1 \leq i \leq N_p\}$
evaluate(P)
for 1 to g **do**
  $P' = \{\}$
  for i = 1 to N_p **do**
    $\mathbf{V}_i$ = mutate($\mathbf{X}_i$)
    $\mathbf{U}_i$ = crossover($\mathbf{V}_i$, $\mathbf{X}_i$)
    $P' = P' \cup \{\mathbf{U}_i\}$
  end for
  evaluate(P')
  P = selection(P, P')
end for
return P_{best}

best linear unbiased prediction (BLUP) on the subset genotype matrix and the phenotype being predicted. The fitness of a particular solution is then the absolute value of the Pearson's correlation between the predicted phenotypes and the true phenotypes. Then, of course, all fitnesses will be in the range [0, 1]. For computational purposes, we dynamically choose between two BLUP strategies: if the number of SNPs in the subset is smaller than the number of samples in the genotype matrix we perform SNPBLUP (see Section 3.1.1.2), otherwise, we perform GBLUP (see Section 3.1.1.3).

3.1.1.1 Random Keys

Differential evolution in its standard form is a real valued optimization technique. Therefore, some accommodation must be made to obtain indices of a feature subset from a vector of real numbers. A well-known technique to accomplish this is known as *random keys* [109]. The random key technique is well-known in EC due to its use in combinatorial optimization tasks such as scheduling [112]. Random keys represent solutions to a combinatorial problem as a real valued vector that is somehow decoded to produce a solution that is always valid in the objective space. This is in contrast to a traditional binary encoding that, when acted on by operators like mutation and crossover, may no longer be a valid solution (e.g., a solution selects more than the desired number of features after

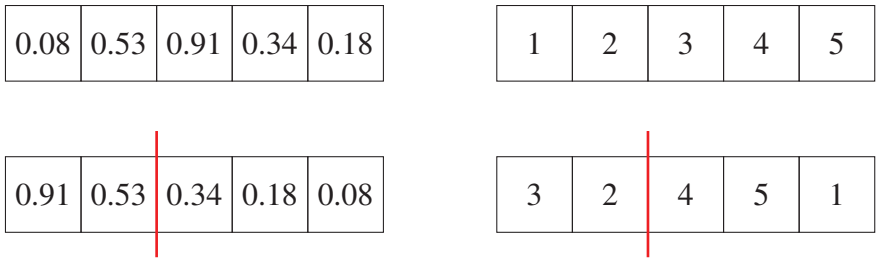


Figure 3.1: An example of the random key decoding used for the DE fitness function. Here, the dimensionality of the problem is 5 and a subset of size 2 is selected from the data matrix. The top vectors represent the unsorted DE vector and its corresponding indices. The bottom vectors are the result of sorting and the red line denotes the choice of the largest two values in the DE vector.

crossover).

To demonstrate random keys, consider the example in Figure 3.1. In this case, imagine the overall problem has a dimensionality of $d = 5$ and the desired subset size is two. In other words, the data matrix $\mathcal{X}$ has five columns, two of which must be selected. Then, each solution in the DE population will be a vector $\mathbf{X}_i \in \mathbb{R}^5$. In order to decode the corresponding feature subset from a given vector, namely $\mathbf{X}_i = [0.08, 0.53, 0.91, 0.34, 0.18]$, it must be sorted to obtain $[0.91, 0.53, 0.34, 0.18, 0.08]$. The choice of descending order is arbitrary—although it is intuitive that larger values in the random key vector are "more important". Then, the indices in the original list that correspond to the two largest values in sorted list are chosen. More concretely, columns 3 and 2 of $\mathcal{X}$ will be used as the subset. If the desired effect occurs, over time DE should increase the random key values of the feature subset indices that are most likely to correlate to a higher fitness. Once a feature subset is obtained, either GBLUP or SNPBLUP is applied using the subset data matrix and the Pearson's correlation between the predicted and true phenotypes is calculated by the following formula

$$r_p = \frac{n\sum_{i=1}^{n} y_i\widehat{y_i} - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} \widehat{y_i}}{\sqrt{\sum_{i=1}^{n} y_i^2 - n\bar{\mathbf{y}}}\sqrt{\sum_{i=1}^{n} \widehat{y_i}^2 - n\bar{\widehat{\mathbf{y}}}}}. \tag{3.1}$$

where $\mathbf{y}$ is the vector of true phenotypes and $\widehat{\mathbf{y}}$ is vector of predicted phenotypes.

3.1.1.2 SNPBLUP

SNPBLUP is a regression directly on the SNP data after adjustment for allele frequencies. More concretely, we use the original data matrix $\mathcal{X} \in \{0, 1, 2\}^{n \times d}$ to construct $\mathbf{p} \in \mathbb{R}^{1 \times d}$, the allele frequency row vector,

$$\mathbf{p} = \frac{\sum_{i=1}^{n} \mathcal{X}_{i,\cdot}}{2n}, \tag{3.2}$$

where $\mathcal{X}_{i,\cdot}$ is the i^{th} row of $\mathcal{X}$. Then $\mathbf{Z} = \mathcal{X} - \mathbf{p}$, where the subtraction here denotes a row-wise subtraction. The SNPBLUP model is then denoted by

$$\mathbf{y} = \mathbf{Z}\mathbf{w} + \boldsymbol{\epsilon} \tag{3.3}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a vector of effect values (weights) and $\boldsymbol{\epsilon}$ is normally distributed noise. It is well know that we can solve for the optimal ridge regression weight vector, $\mathbf{w}^*$, using the Moore-Penrose inverse (pseudoinverse), i.e.

$$\mathbf{w}^* = (\mathbf{Z}^T\mathbf{Z} - \lambda I)^{-1}\mathbf{Z}^T\mathbf{y}, \tag{3.4}$$

where I is the identity matrix. For SNPBLUP, λ is calculated using $\mathbf{p}$,

$$\lambda = \frac{2(1 - h^2)\mathbf{p}(\mathbf{1} - \mathbf{p})^T}{h^2} \tag{3.5}$$

where $\mathbf{1}$ is a vector of ones.

3.1.1.3 GBLUP

GBLUP is mathematically equivalent to SNPBLUP [36] and uses the genomic relationship matrix (GRM), $\mathbf{G} \in \mathbb{R}^{n \times n}$, rather than $\mathbf{Z}$ for prediction. The GRM is again calculated calculated using the allele frequencies $\mathbf{p}$. First, we calculate an intermediate $\mathbf{W}$ for brevity,

$$\mathbf{W} = (\mathcal{X} - \mathbb{1}) - (2\mathbf{p} - \mathbf{1}). \tag{3.6}$$

Here, $\mathbb{1}$ denotes a matrix of ones and the middle subtraction is again row-wise. The GRM is then

$$\mathbf{G} = \frac{\mathbf{W}\mathbf{W}^T}{2\mathbf{p}(\mathbf{1} - \mathbf{p})^T}. \tag{3.7}$$

The regression model then becomes,

$$\mathbf{y} = \mathbf{G}\mathbf{w} + \boldsymbol{\epsilon}, \tag{3.8}$$

which can be solved again using the inverse of $\mathbf{G} + \lambda I$, where $\lambda = (1 - h^2)/h^2$, which exists and will be stable due to the regularization constant. In both SNPBLUP and GBLUP, the datasets used are often manageable enough to compute the appropriate inverses directly. If this were not the case, $\mathbf{w}^*$ can be just as easily obtained with gradient descent since linear regress has a convex loss function.

3.1.2 Mutation

The mutation operator uses a solution vector, $\mathbf{X}_i$, to create a *donor vector*, $\mathbf{V}_i$. This is done through some linear combination of members of the population. There are many proposed methods for this process. The two presented here are known as DE/rand/1,

$$\mathbf{V}_i = \mathbf{X}_a + F \cdot (\mathbf{X}_b - \mathbf{X}_c), \tag{3.9}$$

and DE/current-to-best/1,

$$\mathbf{V}_i = \mathbf{X}_i + F \cdot (\mathbf{X}_{best} - \mathbf{X}_i) + F \cdot (\mathbf{X}_a - \mathbf{X}_b). \tag{3.10}$$

Where a, b, and c are uniformly sampled random integers in the range $[1, N_p]$, $a \neq b \neq c \neq i$, for DE population size N_p, and $\mathbf{X}_{best}$ is the vector with the best associated fitness. Finally, $F > 0$ is a hyperparameter known as the *mutation factor* which can intuitively be seen as how large the "jumps" are in a DE update. DE/rand/1 is the original mutation method presented by Storn and Price [20]. Both methods combine some sort of information from the population to guide the search. Next, the information in the donor vector is shared with its associated solution vector.

3.1.3 Crossover

Crossover uses donor vector $\mathbf{V}_i = \left[v_{1,i}, v_{2,i}, \ldots, v_{d,i}\right]$ and parent vector $\mathbf{X}_i = \left[x_{1,i}, x_{2,i}, \ldots, x_{d,i}\right]$ to create a *trial vector*, $\mathbf{U}_i = \left[u_{1,i}, u_{2,i}, \ldots, u_{d,i}\right]$. There are two methods for this: binomial (or

uniform) and exponential crossover [20]. For this study, only binomial crossover is used and in general exponential crossover is less widely studied[1]. For each entry in the trial vector,

$$u_{j,i} = \begin{cases} v_{j,i} & \text{if } rand[0,1) < C_r \text{ or } j = j_{rand} \\ x_{j,i} & \text{otherwise.} \end{cases} \tag{3.11}$$

Here, $C_r \in [0, 1]$ is the crossover rate hyperparameter, $rand[0, 1)$ is a uniform random number in the half-open range $[0, 1)$, and j_{rand} is a uniform random integer from $[1, d]$ that is generated once per generation for each solution in the population. The purpose of j_{rand} is to ensure at least one index is crossed-over with the donor vector for each vector in the population. This is as defined in [20], however at higher dimensions j_{rand} becomes less necessary.

3.1.4 Selection

The standard selection operator at each generation is a simple tournament selection between $\mathbf{X}_i$ and $\mathbf{U}_i$. If the fitness of $\mathbf{U}_i$ is greater than the fitness of $\mathbf{X}_i$, it replaces $\mathbf{X}_i$ and continues on to the next generation in the i^{th} index of the population. This is a "greedy" feature of DE. Often in evolutionary algorithms, having a better fitness increases a particular solution's chances of continuing to the next generation or creating new solutions, but does not always guarantee it.

3.2 Self-adaptive Differential Evolution

The above description of DE leaves out discussion on tuning the associated hyperparameters F, C_r, and N_p. These values often have a dramatic influence on the convergence of DE [98, 99]. As a result, some effort has gone into alleviating the choice of F and C_r through *self-adaptive* methods that "learn" these values throughout the course of a DE experiment. In the methods presented below, this is done by observing which particular settings create trial vectors that successfully enter the next population.

[1]Exponential crossover is analogous to point based crossover common in GAs. This type of crossover is done to exploit underlying structure in optimization problems.

3.2.1 SaDE

Qin and Suganthan present Self-adaptive Differential Evolution (SaDE) in [98] as a way to learn not only the F and C_r parameters, but which mutation method to use as well. For each individual, a donor vector has probability p of being created with DE/rand/1 and probability $1 - p$ of being created with DE/current-to-best/1. Where p is initialized to be 0.5 and updated by calculating

$$p = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_1 \cdot (ns_2 + nf_2) + ns_2 \cdot (ns_1 + nf_1)}. \tag{3.12}$$

Where ns_1 and nf_1 are the number of trial vectors that were produced with donor vectors from DE/rand/1 that entered the next population (a *success*) and the number that did not (a *failure*), respectively. The values ns_2 and nf_2 have the same definition, but count the number of successes and failures for DE/current-to-best/1. Finally, the first 50 generations of the search do not update p to allow some time for the algorithm to stabilize and learn meaningful success and failure rates [98].

Values of F and C_r are newly generated for each individual solution vector at each generation. F is not learned using any particular scheme in SaDE, and is simply randomly sampled from the normal distribution $\mathcal{N}(0.5, 0.3^2)$, then clipped to fall in the range $(0, 2]$. The authors state that C_r is much more important to the performance of DE and chose to adjust it based on the trajectory of the search [98]. To do so, a C_r is sampled for each index in the population every 5 generations from $\mathcal{N}(C_{rm}, 0.1^2)$. Then, similarly to the mutation strategy, every 25 generations, C_{rm} is recalculated based on the values of C_r that successfully produced trial vectors that entered the next population. This method has proved successful on many test problems, so it will be applied here as well.

3.2.2 MDE_pBX

Islam et al. presented the MDE_pBX method in [99]. The main features of this approach are a new mutation scheme called DE/current-to-gr_best/1, p-Best crossover, and a novel parameter adaption scheme for both F and C_r. As its name suggests, DE/current-to-gr_best/1 is similar to the

DE/current-to-best/1 scheme and is expressed as

$$\mathbf{V}_i = \mathbf{X}_i + F \cdot (\mathbf{X}_{gr\_best} - \mathbf{X}_i) + F \cdot (\mathbf{X}_a - \mathbf{X}_b). \tag{3.13}$$

The only difference from Equation 3.10 is using $\mathbf{X}_{gr\_best}$ instead of $\mathbf{X}_{best}$. Here, the index gr_best is the best solution vector chosen from a random $q\%$ of the population. Islam et al. show that this encourages target solutions to be attracted to *good* solutions but not always the *best* solution in the population, avoiding premature convergence [99].

The crossover scheme employed in MDE_pBX, namely p-best crossover, uses regular binomial crossover with the caveat that the solution vector $\mathbf{X}_i$ is no longer chosen in order of the population to be a the parent vector. Instead, the vector is randomly chosen from the p top-fitness vectors in the population. Then, regular crossover is carried out as described in Equation 3.11. Parameter p is calculated with the following expression

$$p = ceil\left[\frac{N_p}{2} \cdot \left(1 - \frac{g_i - 1}{g}\right)\right]. \tag{3.14}$$

Where N_p is the population size, g_i is the current generation, and g is the total number of generations. This formula results in parent vectors initially being chosen randomly from the entire population and gradually decreases to only the best half of the population.

Finally, MDE_pBX also employs a parameter optimization scheme. A mutation rate factor is sampled for each index in the population, following $Cauchy(F_m, 0.1)$, where F_m is calculated using the previous successful generations of trial vectors. Using the notation in [99], let $F_{success}$ be the set of successful mutation factors, then

$$F_m = w_F \cdot F_m + (1 - w_F) \cdot \sum_{x \in F_{success}} \left(\frac{x^{2/3}}{|F_{success}|}\right)^{2/3}. \tag{3.15}$$

Where $w_F = 0.8 + 0.2 \cdot rand(0, 1)$ is a random weighting parameter. Similarly, crossover rates are chosen from $\mathcal{N}(C_{rm}, 0.1)$ where

$$C_{rm} = w_{C_r} \cdot C_{rm} + (1 - w_{C_r}) \cdot \sum_{x \in C_{r\_success}} \left(\frac{x^{2/3}}{|C_{r\_success}|}\right)^{2/3}. \tag{3.16}$$

and $w_{C_r} = 0.9 + 0.1 \cdot rand(0, 1)$. These update functions are intended to gradually change F_m and C_{rm} through weighting the previous stored values. All parameters here are shown to be good general choices in [99] and Islam et al. show MDE_pBX performs well on the 2005 CEC benchmarks [102] and a trajectory planning problem included in the 2011 CEC benchmarks [103].

3.3 Local Search

Local search is a common addition to EC algorithms to perform optimizations during or after the search. Modern DE implementations often carry out a full search, then finish with local search [98, 99] to fine tune the solution obtained. The local search method presented here is described in Algorithm 2. Local search serves to relax the hyperparameter corresponding to the size of the feature subset to select. Before the search begins we may only have an estimate of how many features to select from the original set. As long as we overestimate, local search can serve to prune out unnecessary features. The search operates by choosing the best solution by fitness, $\mathbf{X}$, and decoding it to obtain feature subset $\mathbf{T} = [t_1, \ldots, t_s]$. Binary mask $\mathbf{m} = [m_1, \ldots, m_s]$ determines which indices of $\mathbf{T}$ to choose. More concretely $\mathbf{T}(\mathbf{m})$ defines a subset of $\mathbf{T}$ according to the following rules,

$$\begin{aligned} m_i = 0 &\implies \text{remove } t_i \\ m_i = 1 &\implies \text{keep } t_i. \end{aligned} \tag{3.17}$$

Local search is intended to determine which indices to retain by iteratively checking if the performance of $\mathbf{T}(\mathbf{m})$ increases when index i is removed. If performance increases, index i is removed permanently.

3.4 Coevolution of Subset Size

To further alleviate hyperparameter choice, one can include the size of the feature subset, s, in the solution vector. This results in a mixed valued vector with entries 1 through d being in $\mathbb{R}$ and entry $d + 1$ being an integer. During the decoding process described in Section 3.1.1.1, the value at entry $d + 1$ is rounded to the nearest integer and used as the subset size. This results in a *coevolution*

Algorithm 2 Knockout Local Search

Input: Final population $\{\mathbf{X}_1, \ldots, \mathbf{X}_{N_p}\}$, population size N_p, desired feature subset size s.
Output: Locally optimal solution $\mathbf{T}(\mathbf{m})$

$\mathbf{X} = \max_{1 \leq i \leq N_p} \text{fitness}(\mathbf{X}_i)$
$\mathbf{T} = \text{decode}(\mathbf{X})$
$f = \text{evaluate}(\mathbf{T})$
$\mathbf{m} = [1, \ldots, 1]$
for $i = 1$ to s **do**
    $m_i = 0$
    $f_{mask} = \text{evaluate}(\mathbf{T}(\mathbf{m}))$
    if $f_{mask} > f$ **then**
        $f = f_{mask}$
    else
        $m_i = 1$
    end if
end for
return $\mathbf{T}(\mathbf{m})$

scheme where both the subset contents and size are both optimized during the search. The simplest solution to this new problem is to include all markers in the subset and do a full GBLUP on the entire genome. As discussed previously, this results in the model fitting to uninformative markers. Therefore, we impose a linear combination of objectives that maximizes accuracy and minimizes subset size. More concretely, the fitness function, J, becomes

$$J(\mathbf{X}_i) = accuracy\_of\_model(\mathbf{X}_i) - \gamma \frac{s}{d}, \tag{3.18}$$

where γ is a weight that must be tuned.

3.5 Seeded Initial Population

In order to incorporate domain knowledge, the results of a GWAS can be included in the initial DE population. Through *seeding*, the indices corresponding to the s most significant SNPs are marked with a value of 1 in some vector in the population. Since all solution vectors are initialized in the range $[0, 1)$, these indices will form the subset for that particular vector. Due to the greedy nature of DE, the search will never reach a fitness value that is worse than the seeded initial vector.

3.6 Heritability Thresholding

As mentioned previously, the estimated heritability of a trait can be used to somehow augment the DE search. Reaching values equal to or greater than h means the search has begun to overfit to the validation set (see Section 3.7.2) since it has gone above the highest possible accuracy. The peculiarity of h should be emphasized. It is quite uncommon for any predictive modeling problem to have a hard threshold for performance. Usually a practitioner shoots for some value that is as high as possible. The methods explored here are an initial look into using this value for preventing overfitting in search based feature subset selection.

3.6.1 Simple Halting

A rudimentary method is to simply stop the search when some statistic of the population reaches $h(1 + \alpha)$ for some small $\alpha \in (-1, 1)$. Where these possible measures could be the maximum, minimum, median, or mean fitness of the population. This method is based on the fact that we can treat h as a hard threshold and the idea that it could be better to simply stop searching rather than continue a search that is already known to be overfit to the validation set.

3.6.2 Marker Removal

Another possibility is to remove some amount of the available SNPs when $h(1 + \alpha)$ is reached. Let $\mathbf{X}_{best}$ be the current best solution vector and its decoded vector be $\mathbf{T}_{best}$. From $\mathbf{T}_{best}$ some r SNPs are removed and combined with the current set of removed SNPs, $\mathbf{R}$. Then, for every fitness evaluation, the decoded vector sent to be evaluated is no longer simply $\mathbf{T}_i$, for solution vector i, but $\mathbf{T}_i \setminus \mathbf{R}$, where $\setminus$ denotes set difference. When this occurs, the current population must also be reevaluated to get new fitnesses based on the removed markers. Any vector that is empty after this difference is simply assigned a fitness of 0. This method is employed to encourage selecting SNPs that are less overfit to the dataset. At the end of the search, when evaluating each vector on the testing set, $\mathbf{T}_i \cup \mathbf{R}$ is evaluated for each i.

3.7 Data

3.7.1 Simulation

To demonstrate the performance of DE, simulated data is favorable to control the complexities introduced in genomic data. Specifically, the number of QTL, desired trait heritability, h^2_{in}, and absence of epistatic effects are controlled for through the following process. Using the genotype matrix $\mathcal{X} \in \{0, 1, 2\}^{n \times d}$, q columns are uniformly chosen the QTL. Let $\boldsymbol{\beta}^* = [\beta_1, \ldots, \beta_q]^T$ be the true, simulated QTL effects. First, $\beta_i \sim \mathcal{N}(0, 1),\ \forall i$. Then, $\boldsymbol{\beta}^*$ is adjusted by calculating the variance of each allele. For diploid organisms, there are three alleles: heterozygous (AB) and homozygous (AA, BB). To determine the genetic variance, we calculate the rate that each allele occurs (i.e. total occurrences of a particular allele divided by total number of sample in $\mathcal{X}$). Then the genotypic variance is simply the variance of these three values, namely V_G. The true genetic values are then calculated by

$$\mathbf{t}_g = \frac{\mathcal{X}_q \boldsymbol{\beta}^*}{V_G h^2_{in} t_v}, \tag{3.19}$$

where $\mathcal{X}_q \in \{0, 1, 2\}^{n \times q}$ is the matrix consisting of the q columns that were chosen to be QTL, t_v is the desired output trait variance, and the division corresponds to an element-wise division of $\mathcal{X}_q \boldsymbol{\beta}^* \in \mathbb{R}^n$. The vector $\mathbf{t}_g$ is then centered at zero by subtracting its mean. Finally, to calculate the actual phenotypes, $\mathbf{y}$, "environmental" noise is added to $\mathbf{t}_g$. This is done by calculating $\mathbf{y} = \mathbf{t}_g + \boldsymbol{\epsilon}$. To calculate $\boldsymbol{\epsilon}$, $\mathbf{e} = \left[e_1, \ldots, e_q\right],\ e_i \sim \mathcal{N}(0, \left[t_v(1 - h^2_{in})\right]^2)$ is first sampled, then

$$\boldsymbol{\epsilon} = \mathbf{e} \cdot \sqrt{\frac{(1 - h^2_{in}) t_v}{var(\mathbf{e})}}. \tag{3.20}$$

When estimated, the heritability of the simulated trait will be approximately equal to the desired heritability, h^2_{in}.

3.7.2 Splitting

Data splitting is an important part of wrapper based feature selection as it can control some overfitting the method displays. Here, the data is split into three groups. First, a testing set is

removed from the data that will not be used in any way during the search. This will serve as external validation for the DE search process to report results. Then, with the remaining data a few choices can be made. The simplest method uses no cross-validation and splits the data again into training and validation. The training set is then used to train the BLUP model in the fitness evaluation and the validation set is used to obtain the prediction accuracy (see Section 3.1.1). However, since the DE search assigned fitness based on the same the same validation set for the entire experiment, it is likely that the search will overfit to the validation set. To combat this, three cross-validation schemes are proposed: (1) *intergenerational* cross-validation which does k-fold cross-validation over the course of the search, changing the validation set at each generation. More concretely, at generation g_i, validation set $k \bmod g_i$ will be used to calculate prediction accuracy, and the of the data used as training. (2) Intragenerational cross-validation performs k-fold cross-validation at every fitness evaluation. This method increases the computation time required by a factor of k. But, intuitively, may provide a more reliable fitness value. (3) Monte Carlo cross-validation uniformly samples a random subset of the data to be used as validation. The intuition behind this method is to drive the search through obtaining solutions that better generalize across the entire validation set.

CHAPTER 4

RESULTS

4.1 Experimental Setup

This results section is split into two main experiments in Sections 4.3 and 4.4. Each presents results outlining the behavior of the baseline DE system and how it improves when certain features are added to the system. These results should be seen as an exploratory study into the behavior of DE for feature selection in genomic prediction. Once the best DE strategy is determined, a completely separate dataset will be used to validate the exploratory results. For clarity, DE with no added components will be referred to as *vanilla DE*, and those with added components will be given appropriate monikers. Section 4.3 presents results for a fixed subset size and Section 4.4 shows results for coevolutionary DE, where the size of the subset is also searched. See Table 4.1 for a list of general evolutionary parameters which apply to the fixed subset and coevolutionary experiments.

All significance testing was performed with the Mann-Whitney U test [113] implemented in the SciPy statistics library [114]. The Mann-Whitney U test is a popular choice for stochastic optimization due to the fact that it has no assumptions on the distribution of the random variables being tested. All accuracy measures—excluding BayesR—were obtained with linear regression, which has normally distributed error. This suggests that a t-test [115] should be used. However, DE obtains a subset of the data stochastically, which has an unknown effect on the distribution of final testing accuracy. Hence, the safer choice of the U test was preferred.

4.2 Data

The data used for both exploratory studies is a population of 7,539 sheep with 48,588 SNP markers. In other words, the genotype matrix, $\mathcal{X} \in \{0, 1, 2\}^{7539 \times 48588}$. These genotypes were not gathered as a part of this study, only analyzed after the fact. Using this genomic data, a

Table 4.1: A table of evolutionary parameters used for DE.

Parameter (symbol)	Value
Generations (g)	5000
Population Size (N_p)	50
Crossover Rate (C_r)	0.8
Mutation Factor (F)	0.5
Replicates	10

phenotype was simulated by the method described in Section 3.7.1 with $q = 100$ QTL and desired heritability $h^2_{in} = 0.4$. The data was split regardless of cross-validation scheme using a 64%/16%/20% train/validation/test split (see Section 3.7.2; the split was a result of a 80%/20% split on the whole dataset, then another 80%/20% split on the remaining non-test set data). Both k-fold cross-validation schemes used $k = 5$. For the comparison methods—namely GWAS, GBLUP, and BayesR—a 5-fold cross-validation scheme was used to present an average accuracy across the dataset. To visualize the structure of the data and results of a simple GWAS, a Manhattan plot is provided in Appendix Figure A.1. From this plot, it is apparent that this simulation process produced data with many markers well above the Bonferroni threshold.

4.3 Fixed Subset Results

In the fixed subset results, a size of 1000 is used across all experiments. Since we later show results on the coevolutionary method that searches the subset size as well as the subset contents, a simple experimental set for only one subset size is presented here rather than exploring many fixed subset possibilities.

4.3.1 Baseline

As a baseline, vanilla DE will be compared to the three following baseline genomic prediction methods using the entire genome: (1) GWAS; a simple genome-wide association study on the data is computed (see Section 2.2). Predictions are then made using the effect values calculated in the single SNP regressions. (2) GBLUP; see Section 3.1.1.3. (3) BayesR; this is considered the

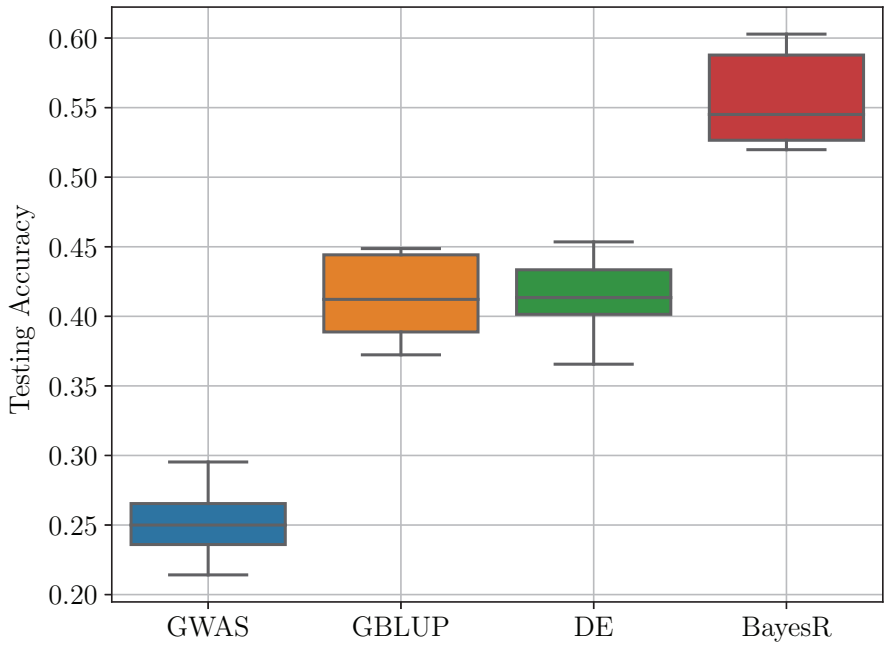


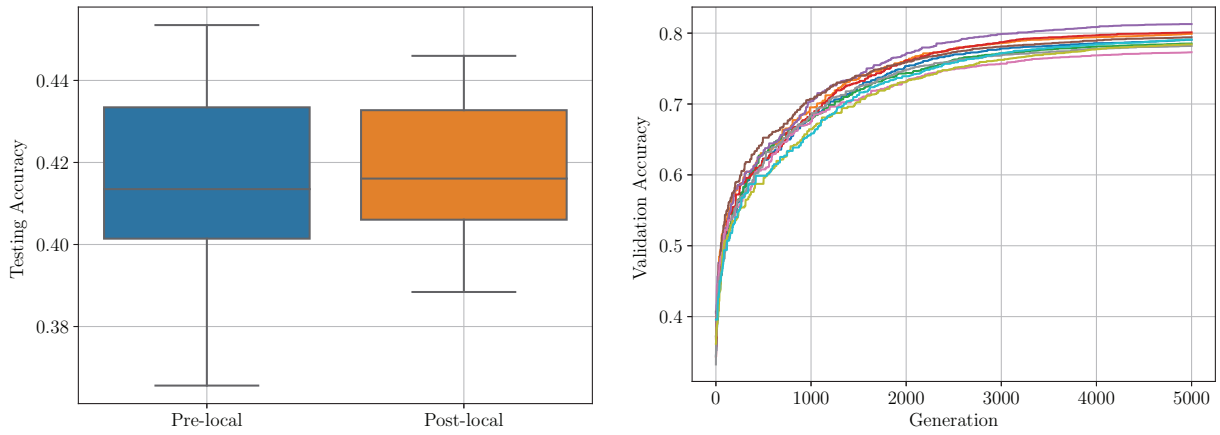
Figure 4.1: Boxplots comparing fixed subset vanilla DE with commonly used genomic prediction methods trained with the entire feature set.

state-of-the-art method in this study. The BayesR experiments use 50,000 iterations with a burn-in of 20,000.

See Figure 4.1 for the results of this baseline study. It is clear that DE alone on this problem was not enough to be competitive with the state of the art method. However, there is something to be said for the comparison against GBLUP. GBLUP is essentially the fitness function for DE, which shows that the same accuracy was obtained using 50× less markers. The GWAS performed poorly in comparison due to the fact that it does not account for any population structure in the prediction and only uses the raw effect values. Now that this baseline has been established, the goal is to reduce the gap between DE and BayesR.

4.3.2 Local Search

The local search results on vanilla DE pre and post-local search are presented in Figure 4.2a. The mean and standard deviation of testing accuracy across the ten replicates for pre-local search are



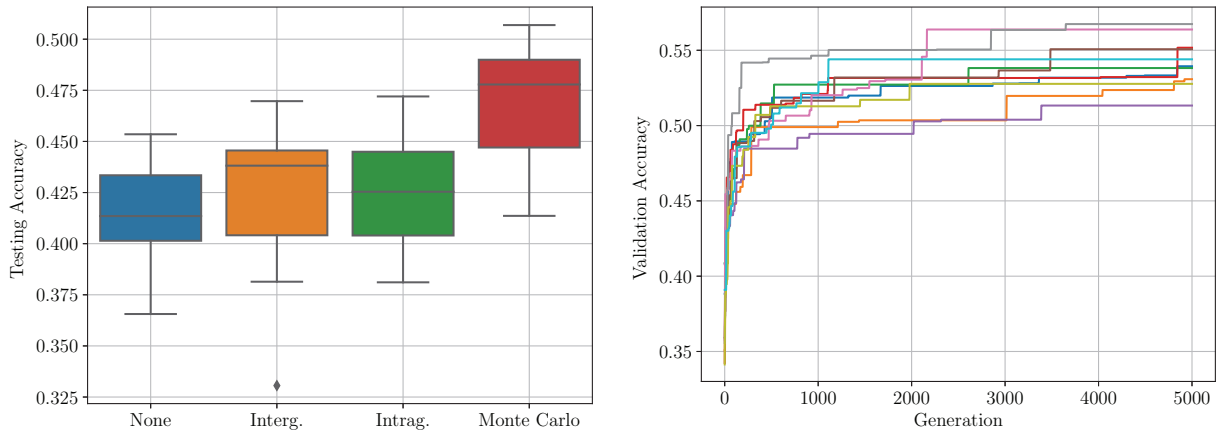
(a) (b)

Figure 4.2: (a) Boxplots comparing the fixed subset vanilla DE experiment before and after the devised local search method. (b) Convergence plot for the maximum fitness at each generation for each replicate in the fixed subset vanilla DE experiment.

0.4232 ± 0.0174, and 0.4180 ± 0.0188 for post-local search. Post-local search is no better than pre-local search with $p = 0.3421$, by the Mann-Whitney U test. The number of features selected after local search was reduced to 753.4 ± 11.7234. This is an uninformative result since it is obvious from Figure 4.2b that DE had likely overfit to the validation set. Since local search is also carried out using the validation data, the result does not change significantly. Reducing subset size while not significantly reducing accuracy is still, however, beneficial since a smaller subset is the goal of this method. Moving forward in results, if a post-local search result showed a statistically significant increase in performance, it will be reported instead of its pre-local search counterpart. See Section 4.3.4 for more results and discussion on overfitting in the fixed subset experiments.

4.3.3 Cross-validation

As described in Section 3.7.2, three cross-validation schemes were compared: intergenerational, intragenerational, and Monte Carlo cross-validation. See Figure 4.3a for boxplots comparing the performance of each method and see Table 4.2 for more detailed results, including significance testing. From Figure 4.3b, it appeared that the Monte Carlo search did not converge and did not cross the h threshold of 0.6325. At this point, there are to confounding factors that seem to govern



(a) (b)

Figure 4.3: (a) Boxplots comparing the different cross-validation schemes for the fixed subset experiments. (b) Convergence plot for the maximum fitness at each generation in the fixed subset Monte Carlo cross-validation experiment.

overfitting: convergence of the search and final validation accuracy. A "close to" converged search will likely have a validation accuracy over h. However, does this mean the search is overfit because it has converged or because its fitness is greater than h? It is also possible that Monte Carlo cross-validation presents a more difficult search problem and more generations are needed for DE to begin to overfit to the validation data.

See Appendix Figure A.2 for the convergence plots of inter and intragenerational cross-validation. These plots both closely resemble the convergence of vanilla DE in Figure 4.2b. This further confounds the above discussion on convergence and final validation accuracy. In the following section a test is presented to untangle these factors. Further evidence of Monte Carlo *not* being overfit to the validation set comes from the local search results. None of the ten replicates showed local search remove *any* of the 1000 features selected during the search. Possibly suggesting that the subset found with this method generalizes better due to the randomization. See Appendix Figure A.3 for an investigation into the convergence of Monte Carlo cross-validation. After 10,000 generations, no significant improvement was observed, suggesting that the Monte Carlo is fairly converged after 5,000 generations, similar to the other cross-validation methods.

Table 4.2: Results for the fixed subset cross-validation strategies. The mean validation and testing accuracy columns are the mean of the maximum of each replicate's population. The p-value column is obtained by using a Mann-Whitney U test to compare the testing accuracy of any other method to Monte Carlo cross-validation, i.e. the alternative hypothesis is $H_a : \mu_{other} < \mu_{monte\ carlo}$. Significant results are bold where the threshold used is $0.05/3 = 0.0167$.

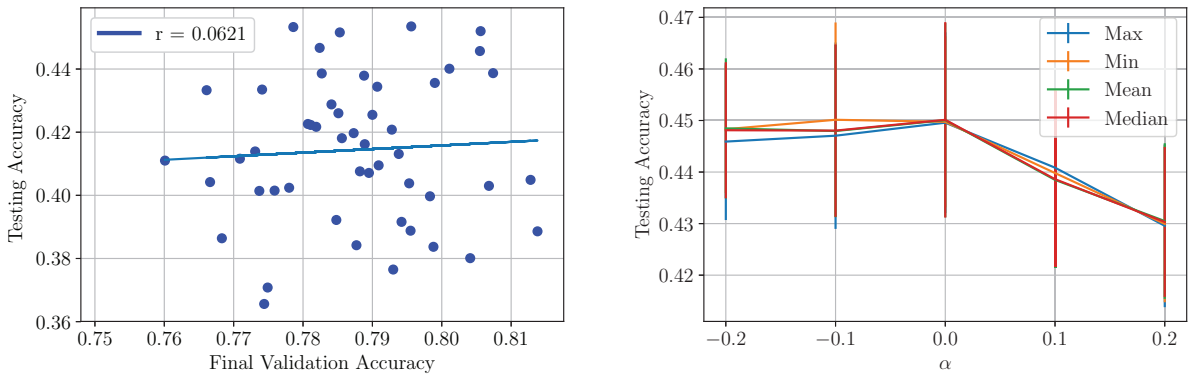
Method	Mean Validation Accuracy ± Standard Deviation	Mean Testing Accuracy ± Standard Deviation	P-value
No cross-validation	0.7913 ± 0.0106	0.4232 ± 0.0174	**0.0**
Intergenerational	0.7108 ± 0.0563	0.4208 ± 0.0388	**0.0041**
Intragenerational	0.6750 ± 0.0066	0.4252 ± 0.0287	**0.0045**
Monte Carlo	0.5427 ± 0.0157	0.4686 ± 0.0284	N/A

4.3.4 Heritability Thresholding

4.3.4.1 Simple Halting

As discussed in Section 4.3.3, there are likely two main confounding factors that contribute to overfitting. To explore this, two experiments were devised: (1) 50 replicates of vanilla DE were run and correlation between final validation and testing accuracies was calculated; (2) the heritability stopping experiments described in Section 3.6 were carried out with $\alpha \in \{-0.2, -0.1, 0, 0.1, 0.2\}$. Meaning the search was stopped when some statistic—mean, median, minimum, or max—of the population's fitness reaches $h(1 + \alpha)$.

See Figure 4.4a for the correlation experiment results. The data clearly showed no correlation between final validation accuracy and testing accuracy. This serves as evidence to conclude that a higher validation accuracy alone did not indicate an overfit search. The convergence of the search was considered next. Figure 4.4b shows the testing accuracy for varying values of α in the heritability threshold formula $h(1 + \alpha)$. For a nonnegative α, the accuracy linearly decreased past $\alpha = 0$. However, the signal was less clear for negative values and no particular fitness statistic seemed give better performance than another. See Table 4.3 for a significance testing on nonnegative α values.



(a) (b)

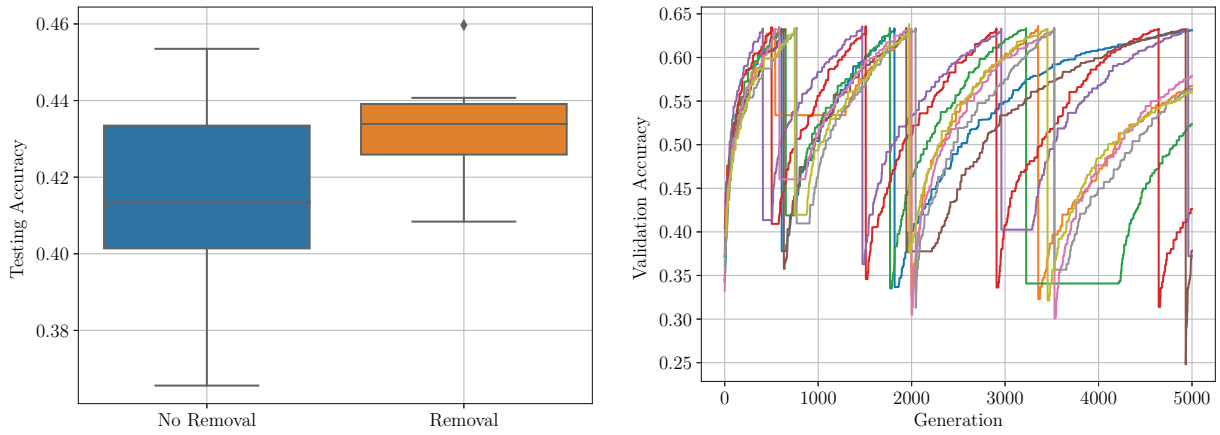
Figure 4.4: (a) A scatter plot with correlation analysis between final validation accuracy and testing accuracy for 50 fixed subset vanilla DE replicates. (b) A plot of the value of alpha used for each stopping condition statistic in the fixed subset experiments, i.e. each search stops when the statistic equals $h(1+\alpha)$. Standard deviation is shown by error bars.

Table 4.3: Detailed results for non-negative values of α in the heritability threshold formula $h(1+\alpha)$ for the fixed subset experiments. The mean testing accuracy column is the mean of the maximum of each replicate's population. The p-value column is obtained by performing a Mann-Whitney U test to compare the testing accuracy of each value of α to $\alpha = 0$, i.e. $H_a : \mu_{other} < \mu_{\alpha=0}$. Bold p-values are significant at a threshold of $0.05/2 = 0.025$. All statistics are gather by combining the results of the mean, median, minimum, and maximum strategy, i.e. each column has 40 samples.

Value of α	Mean Testing Accuracy ± Standard Deviation	P-value
0	0.4499 ± 0.0180	N/A
0.1	0.4394 ± 0.0163	**0.0045**
0.2	0.4301 ± 0.0152	**0.0**

4.3.4.2 Marker Removal

The SNP removal strategy is intended to enforce diversity in solutions by removing features from the possible solution set. See Section 3.6.2 for more details. At each removal, $r = 1000$ SNPs are removed. Meaning the indices selected by the best performing vector are removed entirely. Since this is a less aggressive strategy than halting the search completely, the SNP removal process was performed when the maximum fitness of the population reaches $h \approx 0.6325$. As a result, this experiment did not consider a more stringent study involving α shown in Section 4.3.4.1. Figure 4.5a shows the results of this experiment compared to vanilla DE. The convergence plots in



(a) (b)

Figure 4.5: (a) Boxplots comparing fixed subset vanilla DE and the results of the removal experiment. (b) The convergence plots of each replicate in the marker removal experiment.

Figure 4.5b show the sudden drops in fitness that were caused by reevaluating the entire population after SNPs were removed. The mean testing accuracy of the SNP removal experiment was 0.4322 ± 0.0149 which is better than vanilla DE (0.4232 ± 0.0174; $p = 0.0058$ by the Mann-Whitney U test).

Though SNP removal does seem to be useful, the exact behavior of the subsets obtained and the nature of the search convergence remains unclear. Each SNP removal experiment saw two to four "SNP removal events." Meaning that, at a maximum, the final combined genome size could be 3000 to 5000 SNPs. However, for two, three, and four removal events, the final combined genome length was on average 2,732, 3,673, and 4,121 SNPs, respectively. This may suggest that even with only a few removal events, the most informative SNPs were already discovered toward the beginning of the search. This is supported further by the fact that, as the number of removal events increased, the average combined genome length decreased in respect to its possible maximum. In other words, the SNP removals were less effective as more occurred. The gradual increase in the number of generations required to reach h shown in Figure 4.5b also supports this idea. If all SNPs were equally as informative—which is known to not be the case from the simulation process—it is expected that a convergence behavior seen before the first removal event would be observed after

each removal.

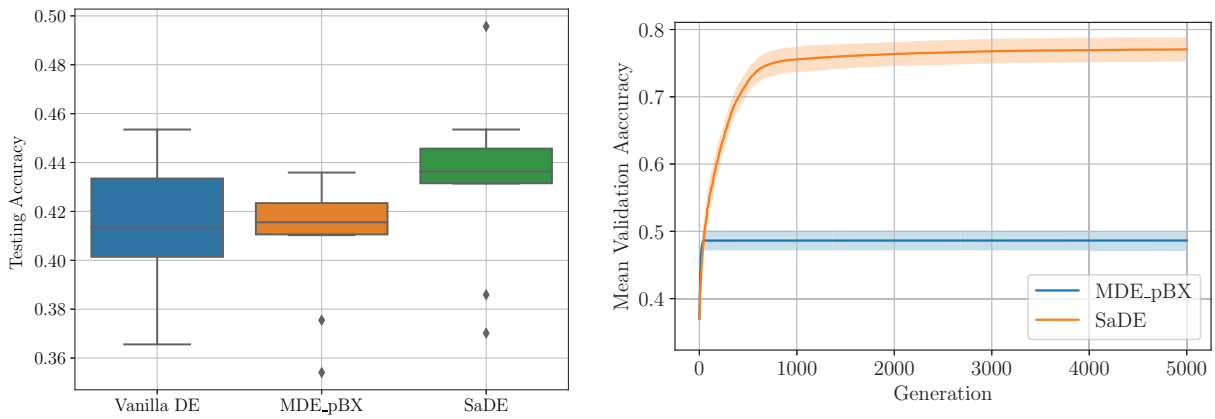
4.3.5 Self-adaptive Differential Evolution

In lieu of a parameter sweep on the mutation factor, F, and crossover probability, C_r, self-adaptive DE methods are preferred. See Section 3.2 for further elaboration on the benefit and efficacy of self-adaptive methods. Though they were proposed as a method to remove hyperparameter choice, SaDE and MDE_pBX do both have hyperparameters, but their values effect the search much less than setting static values for F and C_r [98, 99]. For both methods, the initial values are chosen to be the same as their original settings. Namely, for SaDE the mutation factor distribution was $\mathcal{N}(0.5, 0.3)$ and the crossover rate distribution is $\mathcal{N}(C_{rm}, 0.1)$, where C_{rm} was initially set to be 0.5. Other parameters are the C_{rm} recalculation interval, C_{rm} regeneration interval, and initial learning period which were set to 25, 5, and 50 generations respectively. The initial strategy choice probability, p, was set to 0.5. For MDE_pBX, the mutation factor distribution was set to $Cauchy(F_m, 0.1)$, the crossover distribution followed $\mathcal{N}(C_{rm}, 0.1)$, and the q parameter for determining group size during crossover was set to 15%. The initial values for F_m and C_{rm} were 0.5 and 0.6, respectively.

Table 4.4: Detailed results for the fixed subset self-adpative method experiments. The mean testing accuracy takes the mean of the maximum accuracy of each replicate. The p-value column is obtained by performing a Mann-Whitney U test comparing SaDE to the other two methods, i.e. $H_a : \mu_{other} < \mu_{sade}$. Bold p-values are significant at a threshold of $0.05/2 = 0.025$.

Method	Mean Testing Accuracy ± Standard Deviation	P-value
Vanilla DE	0.4232 ± 0.0174	0.0284
MDE_pBX	0.4093 ± 0.0241	**0.0142**
SaDE	0.4329 ± 0.0329	N/A

See Figure 4.6a for a comparison of vanilla DE to the self adaptive methods. Table 4.4 shows a similar significance testing to previous results under the alternative hypothesis that SaDE performs best. Though the significance testing indicates that MDE_pBX performed significantly worse that SaDE, due to its convergence plot in Figure 4.6b it will be included in further experiments. MDE_pBX showed an unusually fast convergence to a low validation accuracy. This suggests



(a) (b)

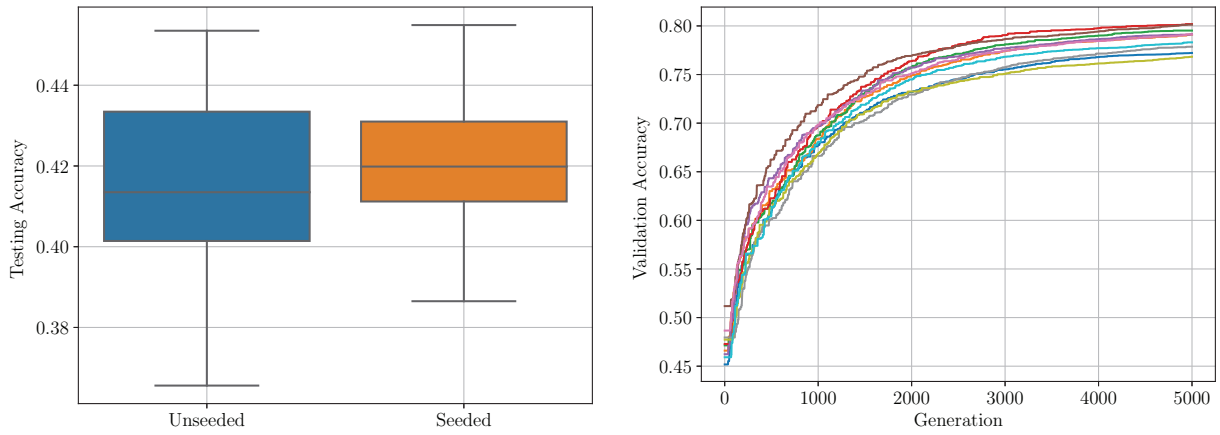
Figure 4.6: (a) Boxplots comparing the self-adaptive methods to vanilla DE in the fixed subset experiments. (b) Convergence plots showing the mean of the maximum validation accuracies for fixed subset MDE_pBX and SaDE. The shaded region shows the standard deviation of the maximum fitness.

the search found a local minimum quickly that it could not escape from. On the other hand, SaDE showed an expected convergence curve, similar to its non-self-adaptive counterpart shown in Figure 4.2b.

4.3.6 Seeding

The seeding process described in Section 3.5 is intended to start the population out at a relatively good neighborhood, rather than a completely random start that is usual of DE—and evolutionary algorithms in general. The GWAS used for seeding was carried out on the remaining internal validation data, i.e. 80% of the data after the 20% testing set was removed. The indices of the most significant effect values were then used to seed the indices of a single individual in the initial population.

See Figure 4.7 for the results of the seeding experiments. It is clear from the convergence plot in Figure 4.7b that seeding gave the search an initial boost in validation accuracy, although this may have simply led to the search overfitting more rapidly as it neared the h threshold. The slight difference in performance shown in Figure 4.7a was not significant using the usual Mann-Whitney



(a) (b)

Figure 4.7: (a) Boxplots comparing seeded and unseeded DE in the fixed subset experiments. (b) Convergence plot showing the fixed subset seeded DE experiment. Note the initial maximum fitness of the population is much higher than the unseeded counterpart in Figure 4.2b.

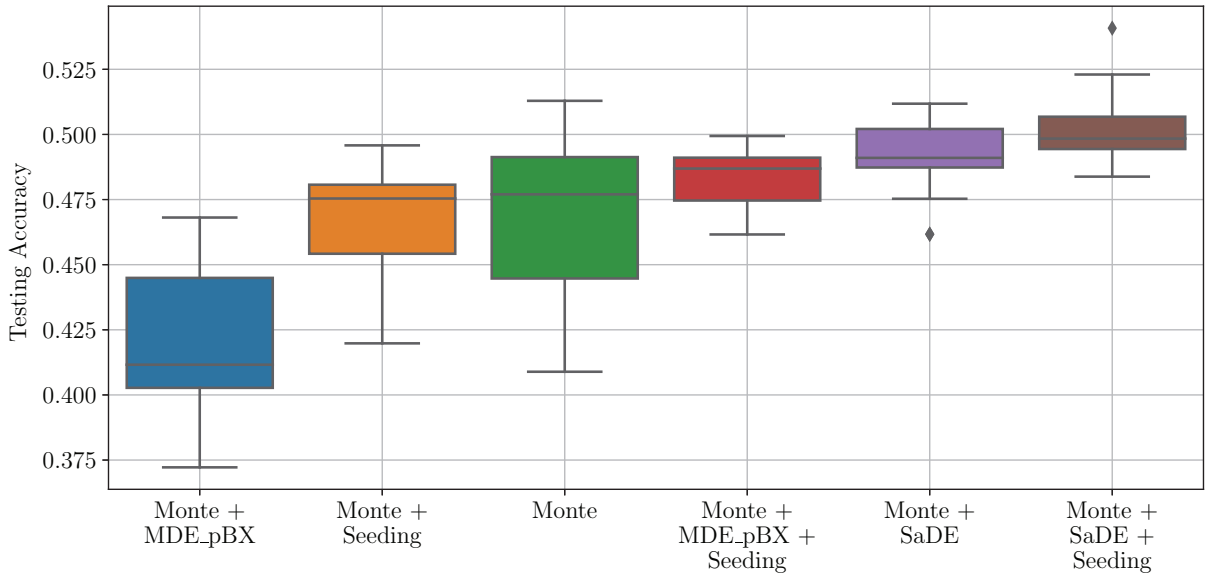


Figure 4.8: Boxplots comparing the results of the fixed subset combination experiments.

U test ($p = 0.1941$). Further experiments that combine components of previous methods will also combine seeding to observe the effect of a warm start.

4.3.7 Combining Components

This section will investigate the effect of combining certain components presented in the preceding experiments. In order to control overfitting in these experiments, it is opted to use Monte Carlo cross-validation instead of a stopping or marker removal scheme. In all experiments presented here, no replicate exceeded the heritability threshold due Monte Carlo cross-validation. Hence, no stopping or removal scheme would have had a chance to be applied. See Figure 4.8 and Table 4.5 for the results of these experiments. Convergence plots for the coevolutionary self-adaptive methods combined with seeding are presented in Appendix Figure A.6.

From these final fixed subset experiments, it is clear that SaDE is the preferred self-adaptive method. The MDE_pBX approach's aggressive convergence seems to be poorly suited for the more difficult problem of feature selection when compared to the results on real parameter optimization in [99]. Furthermore, seeding played a more beneficial role when combined with SaDE and Monte Carlo cross-validation. Figure 4.8 shows that Monte Carlo cross-validation alone did not perform well with seeding. Once it was combined with SaDE, the search found a solution with testing accuracy 0.5408, the best out of all fixed subset experiments.

Table 4.5: Detailed results for the fixed subset component combination experiments. As before, the mean testing accuracy is the average of the maximum testing accuracy of each replicate. Again, as in previous tables, the p-value column is calculated with the Mann-Whitney U test with alternative hypothesis $H_a : \mu_{other} < \mu_{monte\ +\ sade\ +\ seeding}$. Significant results are in bold, using $0.05/5 = 0.01$ as a threshold.

Method	Mean Testing Accuracy ± Standard Deviation	P-value
Monte + MDE_pBX	0.4178 ± 0.0309	**0.0001**
Monte + Seeding	0.4669 ± 0.0225	**0.0010**
Monte	0.4686 ± 0.0284	0.0108
Monte + MDE_pBX + Seeding	0.4822 ± 0.0123	**0.0064**
Monte + SaDE	0.4915 ± 0.0149	0.0766
Monte + SaDE + Seeding	0.5040 ± 0.0171	N/A

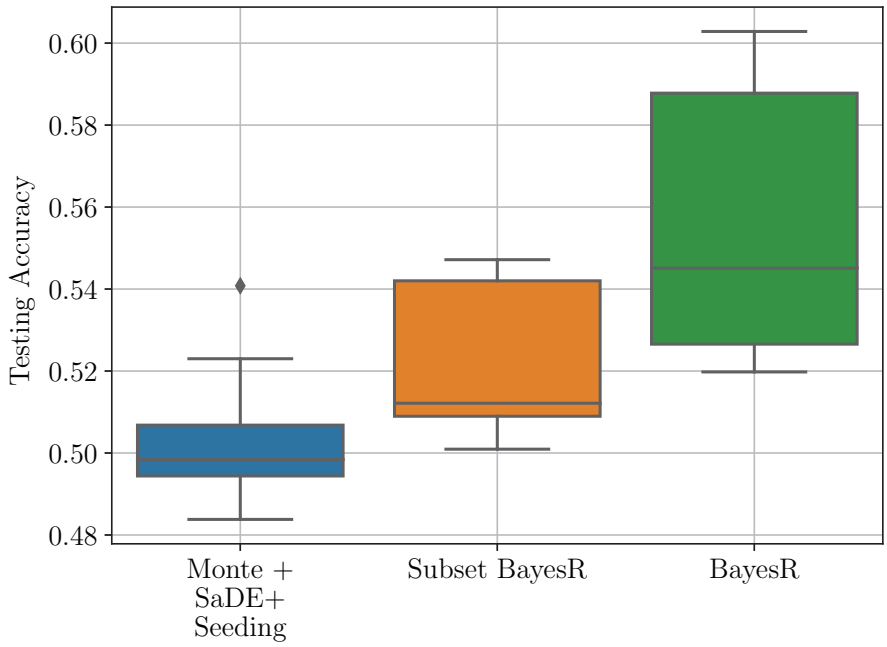


Figure 4.9: Boxplots comparing the results of the best found fixed subset DE method, BayesR trained with the subset found using the best DE method, and standard BayesR trained on the full dataset.

4.3.8 Subset BayesR

BayesR proved to be a more powerful method than DE in the fixed subset experiments. However, due to its computational complexity, it is not suitable for use as an objective function. In an attempt to leverage BayesR in combination with DE, the subset discovered in the DE experiment using Monte Carlo cross-validation, SaDE, and a seeded initial population was used to subset the data before building a model with BayesR. But first, some special considerations were made concerning the data used to train the BayesR model. Specifically, measures were taken to prevent from validating a BayesR model on data that was searched by DE. The BayesR model was trained on the combination of the training and validation datasets used for the DE search and then the accuracy of the model was reported on the DE search's testing dataset. The subset used in the BayesR experiment was chosen by taking the solution with the highest validation accuracy at the last generation. This process was repeated five times among the best five DE experiments based on testing accuracy. Results are presented in Figure 4.9 and Table 4.6. In both cases, standard BayesR

Table 4.6: Detailed results for the subset BayesR experiment. The p-value column was generated by the usual Mann-Whitney U test is performed comparing mean testing accuracy with alternative hypoethsis $H_a : \mu_{other} < \mu_{bayesr}$. Significant results are in bold using a threshold of $0.05/2 = 0.025$.

Method	Mean Testing Accuracy ± Standard Deviation	P-value
Monte + SaDE + Seeding	0.3644 ± 0.0285	**0.0031**
Subset BayesR	0.5222 ± 0.0187	**0.0056**
BayesR	0.5564 ± 0.0332	N/A

performed better than the DE methods. Further investigation and discussion on the performance of BayesR relative to the best found DE methods is presented in Section 4.5.

4.4 Coevolution Results

As mentioned, a preferable alternative to tuning the size of the subset chosen is to also search the subset size. This alleviates a hyperparameter choice, but introduces new complexities discussed in Section 3.4. More specifically, the coevolution fitness function in Equation 3.18,

$$J(\mathbf{X}_i) = accuracy\_of\_model(\mathbf{X}_i) - \gamma \frac{s}{d},$$

introduces the hyperparameter γ that must be tuned before repeating the above experiments. The rest of this section is organized the same as the fixed subset experiments, with the addition of a section presenting the results on tuning γ. Across all coevolution studies, the initial subset size f_i was sampled uniformly from range $[90, 110]$.

4.4.1 Tuning Gamma

An unexpected complexity that arose during tuning γ was the memory requirement that unbounded subset sizes presents. If γ was too small, not enough penalty was applied to the subset size and the search tended toward using the entire marker set in the fitness evaluation. Evaluating individuals in parallel causes the memory limits of the machine used for this study to be met very quickly. The results presented in Figure 4.10 show replicates before this memory limit arose, and have a weak

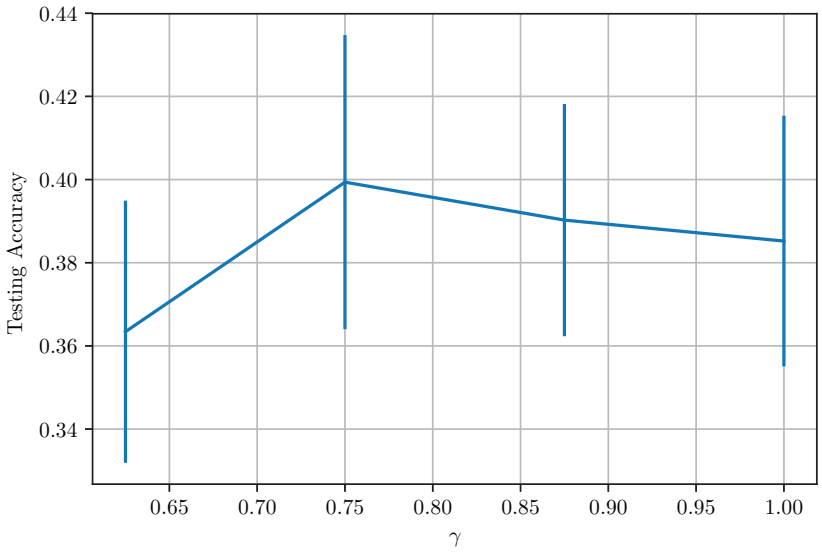
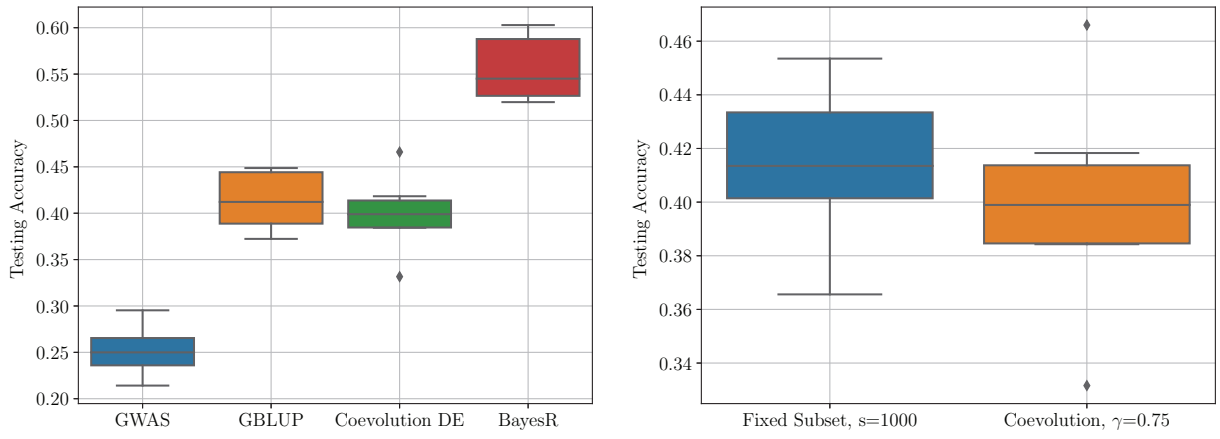


Figure 4.10: Results of the gamma tuning experiment, excluding experiments that failed due to memory limitations.

Table 4.7: Detailed results of the gamma tuning experiments. The mean testing accuracy column takes the average over the maximum testing accuracy in each replicate. As before, the p-value column is obtained with a Mann-Whitney U test comparing the testing accuracy of $\gamma = 0.75$ to the other values of γ, i.e. $H_a : \mu_{other} < \mu_{\gamma=0.75}$. Bold p-values are significant at a threshold of $0.05/3 \approx 0.0167$

Value of γ	Mean Testing Accuracy ± Standard Deviation	P-value
0.625	0.3634 ± 0.0315	0.0306
0.75	0.3994 ± 0.0354	N/A
0.875	0.3903 ± 0.0279	0.3257
1.0	0.3852 ± 0.0302	0.2067

suggestion that $\gamma = 0.75$ may have been preferable to a smaller γ. The values of γ used for this study were 1.0, 0.875, 0.75, 0.625, 0.5, 0.375, and 0.25. See Table 4.7 for detailed results and significance testing. Though $\gamma = 0.75$ was not significantly better than any other value of γ, 0.75 was chosen to move forward with the coevolution study.



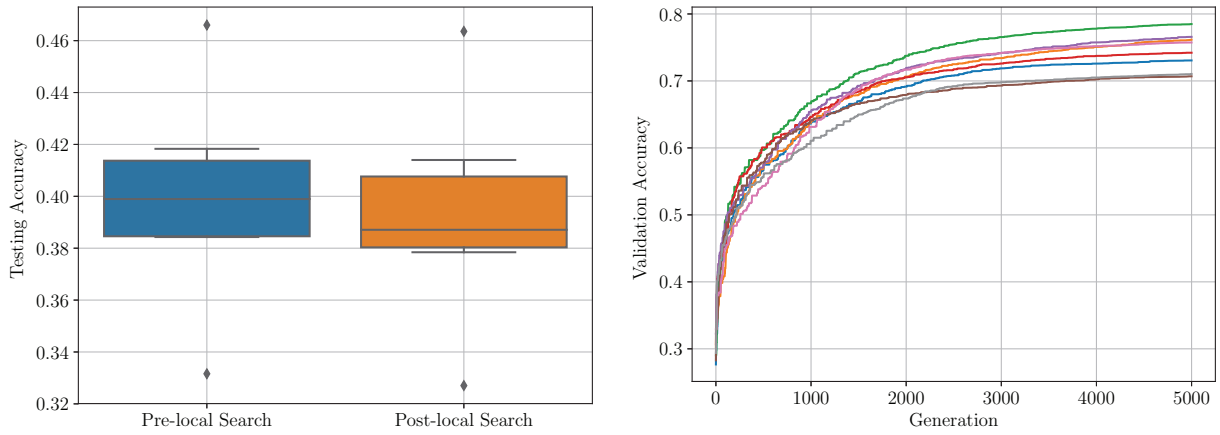
(a) (b)

Figure 4.11: (a) Boxplots comparing coevolutionary DE against commonly used genomic prediction methods using the entire feature set. (b) Boxplots comparing coevolutionary and fixed subset vanilla DE.

4.4.2 Baseline

In Figure 4.11a the same baselines are compared to the coevolution approach as in Figure 4.1. Each experiment keeps track of the average subset size of the population. The mean and standard deviation of the final subset size across each replicate of this experiment was 526.63 ± 158.35. The subset size was quite variable, and was likely tied to the overall variability of the mean testing accuracy displayed in Figure 4.11a.

Similar to the fixed subset results, coevolutionary DE alone was not competitive with BayesR. However, coevolutionary DE obtained results on par with GBLUP, using (on average) 100× less features. Figure 4.11b compares fixed subset and coevolutionary DE. The fixed subset results showed that DE alone was quite variable without any operators to control overfitting. Hence, the comparison of the two basic methods is not very informative. See Section 4.5 for a more "fair" comparison of the two search methods that includes components to control overfitting.



(a) (b)

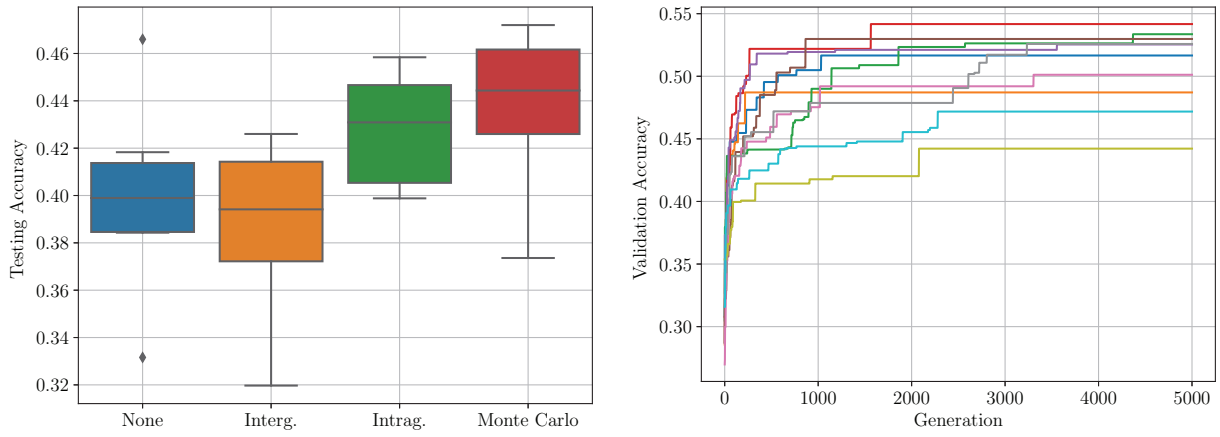
Figure 4.12: (a) Boxplots comparing the testing accuracy pre and post-local search for coevolutionary DE. (b) The convergence plot for eight replicates of coevolutionary DE (two failed due to memory limits).

4.4.3 Local Search

Local search was performed using the same technique for the fixed subset approach described in Section 3.3. Figure Figure 4.12a shows the results of pre and post-local search for coevolutionary DE. This, combined with the results in Figure 4.2 provide clear evidence that the local search method presented in 3.3 did not perform well. Through a feature selection lens, this does make sense. The original motivation for such a simple method was to remove noisy SNPs from the fixed subset method that were only chosen because there was no room to *not* choose them. However, this method was much too greedy, a concept that is well known and pointed out in original sequential feature selection literature such as [81]. As in the fixed subset experiments, if a better result is obtained with local search, it will be reported.

4.4.4 Cross-validation

Cross-validation results paralleling the fixed subset experiments for the coevolutionary approach are presented in this section. Figure 4.13a shows the relative performance of each cross-validation method. Monte Carlo showed a marginal improvement over intragenerational cross-validation.



(a) (b)

Figure 4.13: (a) Boxplots comparing the different cross-validation methods for the coevolution experiments. (b) The convergence plot for the coevolutionary approach with Monte Carlo cross-validation. Note the similarity to Figure 4.3b.

Furthremore, Monte Carlo cross-validation in the coevolutionary setting (Figure 4.13b) showed a similar convergence plot to the fixed subset case in Figure 4.3. Significance testing is presented in Table 4.8. Table 4.8 also displays the mean length of the experiments. Interestingly enough, the better performing methods ended with a shorter mean length and smaller standard deviation across the replicates. The convergence plots for inter and intragenerational cross-validation appear similar to the vanilla DE convergence plot in Figure 4.2b and are presented in Appendix Figure A.5.

Table 4.8: Table showing detailed coevolutionary DE results for each cross-validation experiments. The mean accuracy columns take the average of the maximum validation/testing accuracy for each replicate. The average length column refers to the average of the highest fitness individuals at final generation of each replicate. The p-value column is calculated with the usual significance testing using the Mann-Whitney U test with alternative hypothesis $H_a : \mu_{other} < \mu_{monte\ carlo}$. Significant results are shown in bold using a threshold of $0.05/3 \approx 0.0167$.

Method	Mean Val. Acc. ± Stdev	Mean Test. Acc. ± Stdev	Mean Length ± Stdev	P-value
No cross-validation	0.7450 ± 0.0258	0.3994 ± 0.0354	656.4 ± 249.9	**0.0117**
Intergenerational	0.6755 ± 0.0450	0.3859 ± 0.0363	438.1 ± 199.9	0.0141
Intragenerational	0.5804 ± 0.0304	0.4274 ± 0.0227	265.1 ± 72.6	0.1309
Monte Carlo	0.5075 ± 0.0301	0.4399 ± 0.0278	266.3 ± 95.4	N/A

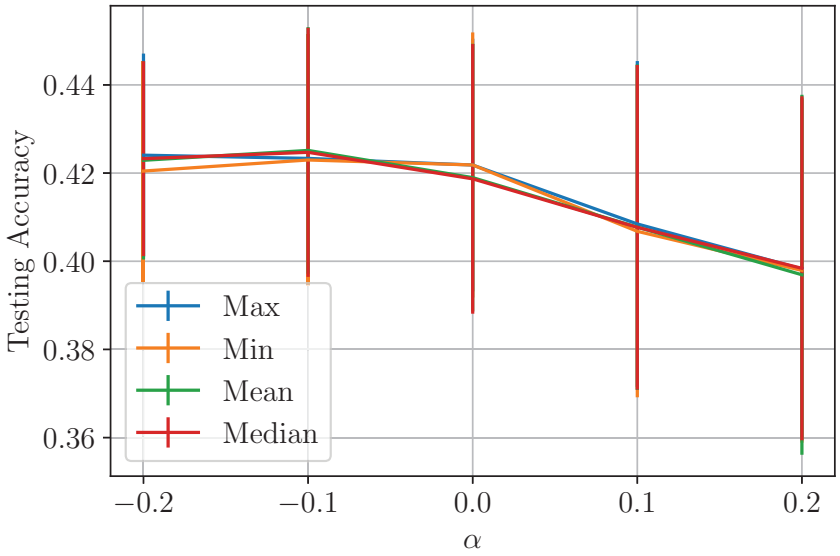


Figure 4.14: A plot of the value of alpha and mean testing accuracy for each stopping condition statistic in the coevolutionary DE experiments. The search halted when a given statistic reaches $h(1 + \alpha)$. Standard deviation is shown by error bars.

4.4.5 Heritability Thresholding

4.4.5.1 Simple Halting

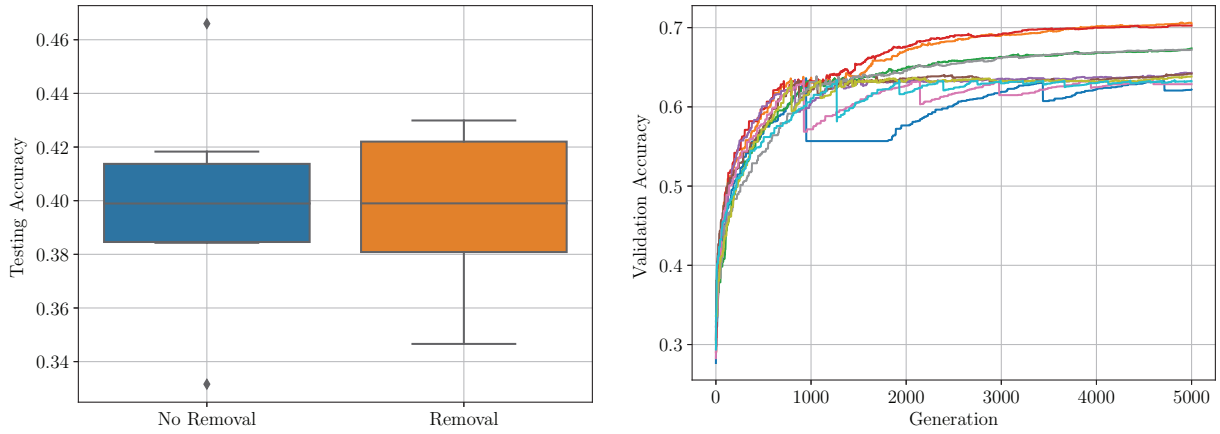
The same analysis of α is presented in Figure 4.4b is repeated for the coevolutionary setting to investigate the effect of stopping a search early on testing accuracy. See Figure 4.14 for the results of this experiment. Again, there was no appreciable difference between each population statistic, as well as no real difference when using a negative alpha. Detailed results and significance testing for the alpha tuning experiments are presented in Table 4.9. Achieving essentially the same result in both fixed and coevolutionary DE shows that the h threshold is an important metric. Search methods not applying some means to stay below h are likely to overfit

4.4.5.2 Marker Removal

Marker removal for the fixed subset case simply removed s—the size of the subset—markers from the "allowed" feature set at each removal. Since this subset size changes in the coevolution case,

Table 4.9: Detailed results for non-negative values of α in the heritability threshold formula $h(1+\alpha)$ in the coevolutionary experiments. The mean testing accuracy column is the mean of the maximum of each replicate's population. The p-value column is obtained by performing a Mann-Whitney U test to compare the testing accuracy of each value of α to $\alpha = 0$, i.e. $H_a : \mu_{other} < \mu_{\alpha=0}$. Bold p-values are significant at a threshold of $0.05/2 = 0.025$. All statistics are gather by combining the results of the mean, median, minimum, and maximum strategies, i.e. each column has 40 samples.

Value of α	Mean Testing Accuracy ± Standard Deviation	P-value
0	0.4203 ± 0.0230	N/A
0.1	0.4077 ± 0.0371	0.03830
0.2	0.3979 ± 0.0393	**0.0044**

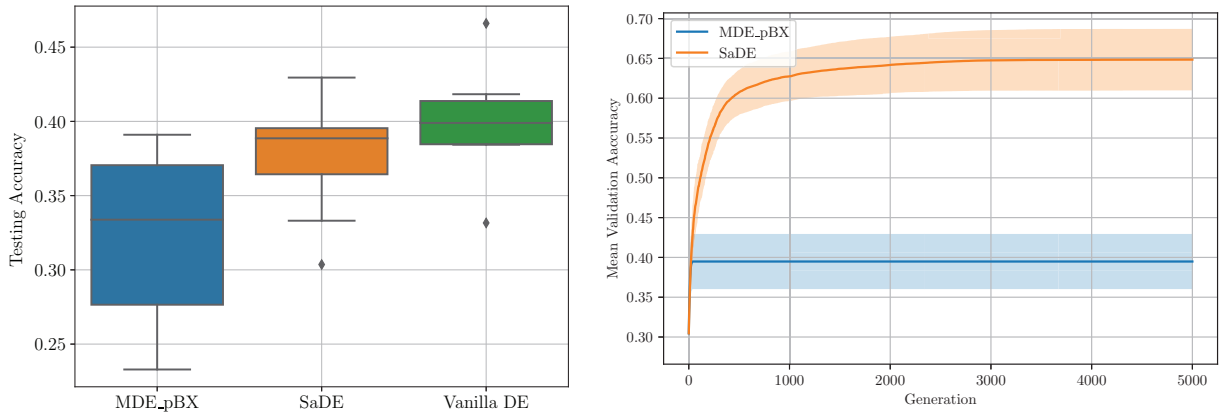


(a) (b)

Figure 4.15: (a) Boxplots comparing the testing accuracy of the coevolutionary marker removal experiments. (b) The convergence plot for the coevolutionary marker removal experiment.

each removal simply takes the away entire subset of the best individual when the maximum fitness of the population reaches h. Again, since this is a less aggressive strategy than stopping the search, a study on α is forgone to focus on the simpler case of $\alpha = 0$.

Figure 4.15 shows the results of the marker removal experiments for the coevolutionary setting. The mean testing accuracy for the removal experiment was 0.3965 ± 0.0279. In comparison to the vanilla DE run, there was no significant difference between the replicates ($p = 0.5354$). Figure 4.15b shows the convergence plot for the removal experiment. This plot may explain the lack of performance compared to the fixed subset experiments. Compared to the fixed subset case



(a) (b)

Figure 4.16: (a) Boxplots comparing the self-adaptive methods to vanilla DE in the coevolutionary case. (b) Convergence plots showing the mean of the maximum validation accuracies for coevolutionary MDE_pBX and SaDE. The shaded region shows the standard deviation of the maximum fitness.

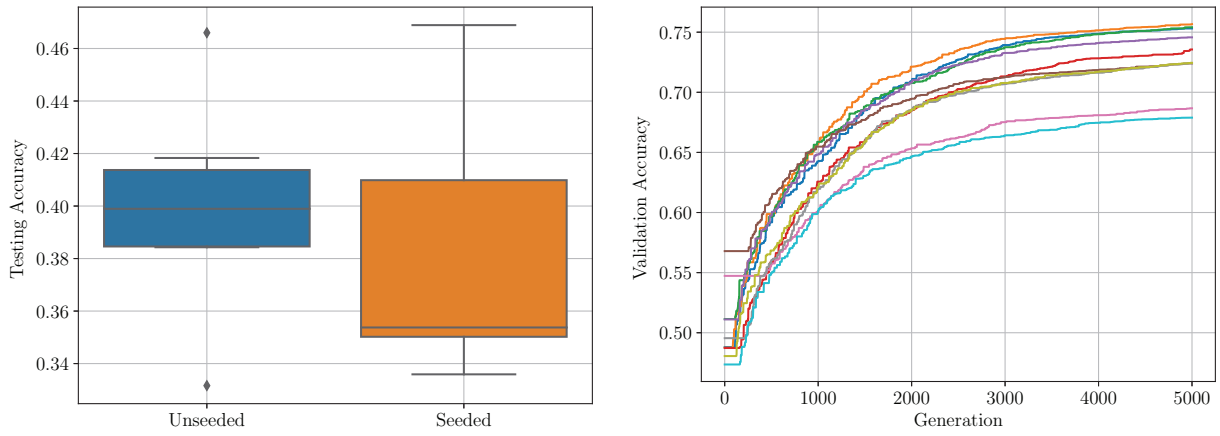
in Figure 4.5b, a marker removal event was much less disruptive to the convergence of the search, and in some cases does not disrupt the search at all. This causes a SNP removal to happen every generation, which becomes useless over time. The average final length across each replicate was 1960.5 ± 903.8—quite a large variance. The maximum total length from any run was 2929.

4.4.6 Self-adaptive Differential Evolution

The coevolutionary experiments used the same parameter distribution initialization as described in Section 4.3.5. The results of these experiments are shown in Figure 4.16 and Table 4.10. Compared to the fixed subset experiments, these methods were much more variable and unreliable. Figure 4.16b shows MDE_pBX converged to a local optimum very early in its search, which follows with the fixed subset case as well. From Table 4.10 it is clear that a self-adaptive method alone was not enough to reach a testing accuracy that is competitive with even vanilla DE.

Table 4.10: Detailed results for the coevolution self-adaptive experiments. The mean testing accuracy column is obtained by taking the mean of the maximum testing accuracy of each replicate. As in previous tables, the p-value column was generated with the Mann-Whitney U test with alternative hypothesis $H_a : \mu_{other} < \mu_{vanillade}$. Significant results are in bold using threshold $0.05/2 = 0.025$

Method	Mean Testing Accuracy ± Standard Deviation	Mean Length ± Standard Deviation	P-value
MDE_pBX	0.3237 ± 0.0524	100.0 ± 4.9	**0.0019**
SaDE	0.3791 ± 0.0367	208.1 ± 59.1	0.1754
Vanilla DE	0.3994 ± 0.0354	656.4 ± 249.9	N/A



(a) (b)

Figure 4.17: (a) Boxplots comparing the seeded and unseeded coevolutionary experiments. (b) Convergence plot of the seeded coevolutionary experiment.

4.4.7 Seeding

The coevolutionary experiments used the same seeding process as the fixed subset experiments, i.e. indices were set to 1 based on GWAS results. However, since there was a small element of random initialization, when a vector was chosen to seed, the number of indices seeded was based on its randomly initialized length. This was the only change when compared to the fixed subset experiment.

Figure 4.17 shows the results of the coevolution seeding experiments. The mean testing accuracy for the seeded results was 0.3817 ± 0.0439, which is less than the unseeded results (0.3994 ± 0.0354), but not significantly ($p = 0.1993$). It seems apparent that the coevolution

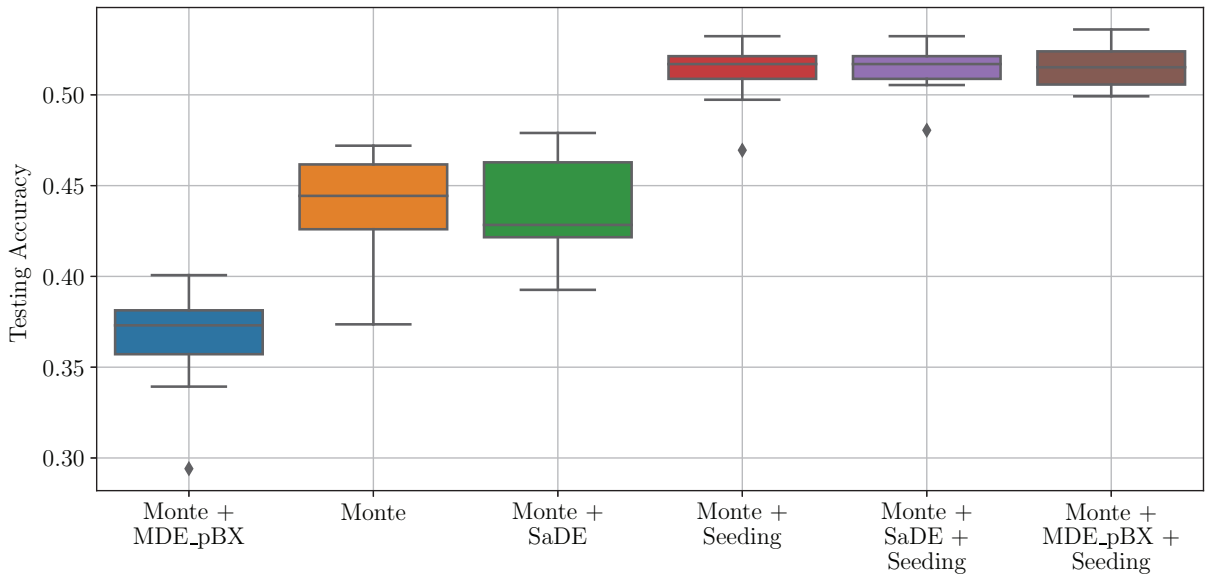


Figure 4.18: Boxplots comparing the results of the coevolution combination experiments.

scheme was more fragile to the overfitting spurred on by seeding. Similar to the fixed subset case, without any measures to prevent the search from overfitting, the final performance of the method suffered.

4.4.8 Combining Components

The same analysis presented in the fixed subset experiments is provided here for the coevolutionary experiments as well. Figure 4.18 shows the results of these experiments. It is apparent that the discussion on seeding and overfitting in Section 4.4.7 holds true. When a method to control overfitting was applied, seeding outperformed unseeded methods. Table 4.11 shows a more detailed result table, with significance testing under the assumption that the method using Monte Carlo cross-validation, MDE_pBX, and seeding was the best performing method. In reality, there was no real difference between the self-adaptive methods and the vanilla DE alternative.

Table 4.11: Detailed results for the coevolutionary component combination experiments. As before, the mean testing accuracy is the average of the maximum testing accuracy of each replicate. Again, as in previous tables, the p-value column is calculated with the Mann-Whitney U test with alternative hypothesis $H_a : \mu_{other} < \mu_{monte + mde\_pbx + seeding}$. Significant results are in bold, using $0.05/5 = 0.01$ as a threshold.

Method	Mean Testing Accuracy ± Standard Deviaiton	Mean Length ± Standard Deviation	P-value
Monte + MDE_pBX	0.3644 ± 0.0285	100.5 ± 4.4	**0.0001**
Monte	0.4399 ± 0.0278	266.3 ± 95.4	**0.0001**
Monte + SaDE	0.4362 ± 0.0273	112.6 ± 9.3	**0.0001**
Monte + Seeding	0.5115 ± 0.0166	132.4 ± 42.2	0.3957
Monte + SaDE + Seeding	0.5136 ± 0.0134	102.9 ± 7.8	0.5451
Monte + MDE_pBX + Seeding	0.5159 ± 0.0116	97.7 ± 4.7	N/A

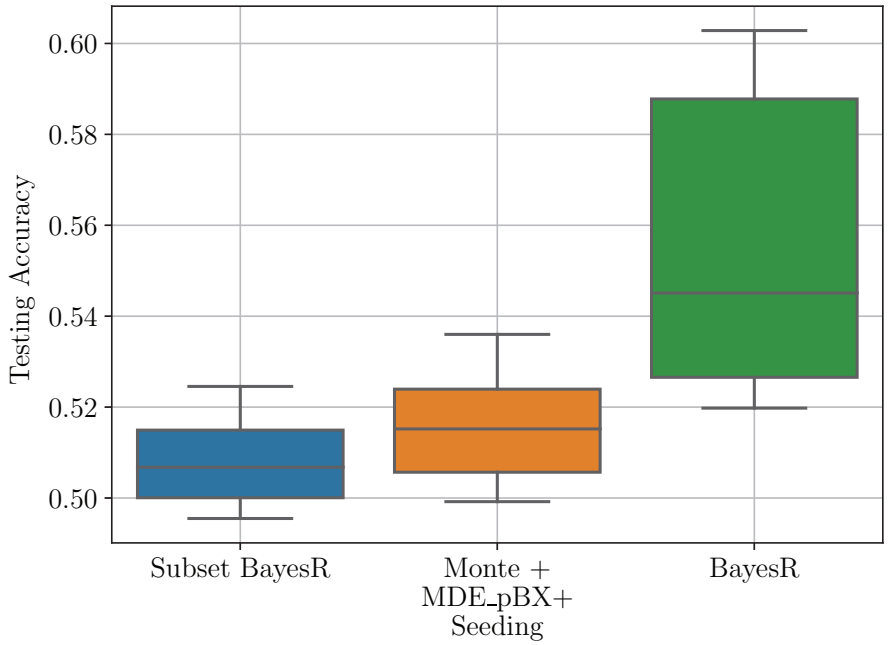


Figure 4.19: Boxplots comparing the results of BayesR trained with the subset found using the best DE method, the best found coevolutionary DE method, and standard BayesR trained on the full dataset.

Table 4.12: Detailed results for the coevolutionary subset BayesR experiment. As before, the p-value column was generated with the Mann-Whitney U test by comparing the mean testing accuracies with alternative hypothesis $H_a : \mu_{other} < \mu_{bayesr}$. Significant results at the threshold $0.05/2 \approx 0.025$ are in bold.

Method	Mean Testing Accuracy ± Standard Deviaiton	P-value
Subset BayesR	0.5068 ± 0.0104	**0.0108**
Monte + MDE_pBX + Seeding	0.5159 ± 0.0116	**0.0117**
BayesR	0.5564 ± 0.0332	N/A

4.4.9 Subset BayesR

The same subset experiment carried out in Section 4.3.8 is presented here for the coevolution study as well. Here, the "best" identified strategy used to obtain a subset was DE with Monte Carlo cross-validaiton, MDE_pBX, and seeding. Results are presented in Figure 4.19 and Table 4.12. In the coevolution case, the DE subset reduced the accuracy of BayesR, furthering the suggestions made in the fixed subset case, see Section 4.3.8.

4.5 System Validation

The preceding results investigated the behavior of feature selection with DE for genomic prediction on a specific simulated dataset. To validate these results, it is necessary to begin again with a new dataset. Results on new data will ensure the best determined methods are not somehow only suited to the simulated data. The genotypes used for this experiment are the same as before: a population of 7,539 sheep with 48,588 SNPs. However, a *real* phenotype was used for validation. This phenotype is called worm egg count (WEC). WEC is a proxy for an animal's immune resistance to parasites and is not very heritable. The heritability of WEC was estimated using a restricted maximum likelihood method—implemented in the NAM R package [116]—to be $h^2 \approx 0.1600$. Hence, the theoretical limit on accuracy is $h \approx 0.4000$. Furthermore, WEC is difficult to predict due its polygenic nature illustrated in the results of a GWAS shown in Appendix Figure A.7 where only two SNPs were identified as significant. It should be noted that the phenotypes were preexisting

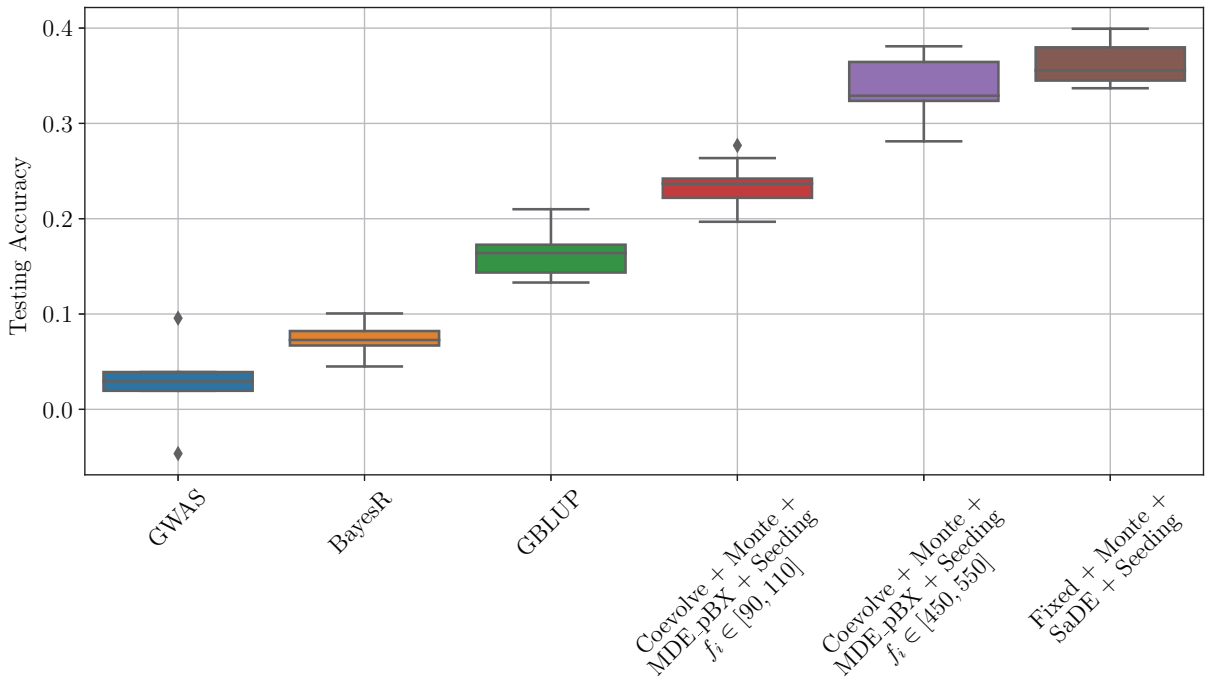


Figure 4.20: Boxplots showing the results of the system validation experiments. Note that f_i is the randomly initialized value of the subset size for each vector $\mathbf{X}_i$.

and not gathered as a part of this study.

To validate the "best" strategies identified in the simulated environment, one strategy from the fixed subset study—SaDE with Monte Carlo cross-validation and seeding—and one from the coevolutionary study—MDE_pBX with Monte Carlo cross-validation and seeding—were applied to the WEC data. No hyperparameters were changed in either case. See Figure 4.20 for the results of this validation experiment and Table 4.13 for the usual detailed analysis. First, the initial coevolution experiment was relatively unsuccessful in comparison to the simulated environment. Under suspicion that the initial subset size was too small, the initial range was changed to [450, 550] and better results were observed. Further investigation into subset size initialization may lead to even better results, though the current result set is close to the theoretical maximum.

All DE methods performed much better than the common genomic prediction methods. The maximum obtained accuracy in the fixed subset validation experiment was 0.3993, which is on par with the estimated theoretical maximum of 0.4. However, this does not suggest DE is always better

than the common methods in a non-simulated environment. As was pointed out previously, the trait being predicted is not very heritable and has only two significant SNPs identified by a GWAS (see Figure A.7). Because of this, it is likely that the performance of BayesR suffered greatly in comparison to the simulated environment where the effects of the QTL—and markers in linkage disequilibrium with the QTL—were large and well defined. That said, these results do suggest that DE based feature selection may be better at identifying marker subsets that better capture relationships between animals, while removing noisy markers.

Table 4.13: Detailed results of the system validation experiments. The p-value column is generated with the usual U test on the mean testing accuracy with alternative hypothesis $H_a : \mu_{other} < \mu_{fixed\ +\ monte\ +\ sade\ +\ seeding}$. Note that f_i is the randomly initialized value of the coevolution subset size for each vector, $\mathbf{X}_i$. Significant results are in bold using a threshold of $0.05/5 = 0.01$

Method	Mean Testing Accuracy ± Standard Deviation	P-value
GWAS	0.0274 ± 0.0454	**0.0013**[1]
BayesR	0.0735 ± 0.01825	**0.0013**[1]
GBLUP	0.1646 ± 0.0267	**0.0013**[1]
Coevolve + Monte + MDE_pBX + Seeding $f_i \in [90, 110]$	0.2353 ± 0.0223	**0.0001**
Coevolve + Monte + MDE_pBX + Seeding $f_i \in [450, 550]$	0.3384 ± 0.0290	0.032
Fixed + Monte + SaDE + Seeding	0.3631 ± 0.0213	N/A

[1]These p-values appear much larger due to the fact that they were generated with only 5 samples from 5-fold cross-validation. In reality a p-value much lower than 0.0013 is appropriate.

CHAPTER 5

CONCLUSION

The preceding results have shown that DE alone can reliably select feature subsets that perform on par with—and in some cases better than—common linear methods. Most searches converge fairly quickly, avoiding the computational strain imposed by more sophisticated genome-wide prediction methods like BayesR. When more operators are added to the search, DE becomes a reliable feature selection for genomic prediction in general. When validated on real data, the best performing DE systems presented here outperform existing methods and approach the neighborhood of maximum theoretical performance. Combining various components such a seeding highlights the potential to apply domain knowledge to DE—and in general, most EC methods.

The most effective applications of domain knowledge in this study were seeding and heritability thresholding. Seeding is a simple process that lends itself well to DE since it begins the search from a good starting point. However, the relationship of the search to h is more nuanced. Results were explored that observed simply stopping a search when validation accuracy reaches h is better than letting the search continue its trajectory. This idea may be beneficial in other applications of genomic prediction as well. It is likely that future search based methods involving genomic prediction will benefit from taking advantage of this unique value.

Adding to the nuance of DE and the theoretical accuracy is the performance of Monte Carlo cross-validation. For the simulated data set, this simple technique limited the convergence of all search methods below h. At a high level, this cross-validation process encourages generalization of the search as a whole. Where simple blocks of validation sets in k-fold schemes can be overfit to quickly, there is not a clear pattern in validation sets to exploit in the Monte Carlo case. Exactly how this relates to overfitting and the value h is likely a fruitful topic for future research.

Genomic prediction has had a profound impact on the agricultural industry in the past and is slated to become a cornerstone in personalized medicine as research involving genetic risk progresses. As complex non-linear methods become popular in bioinformatics and biomedicine,

a precise and readable result provided by a method like DE could lead to a more complete understanding of the complex genotype-pheotype mapping. This study has presented an investigation into the behavior of one simple search method, though avenues for more complex EC and machine learning approaches are endless. The code for this method is available at `https://github.com/ianwhale/tblup`.

APPENDICES

APPENDIX A

A.1 Further Results

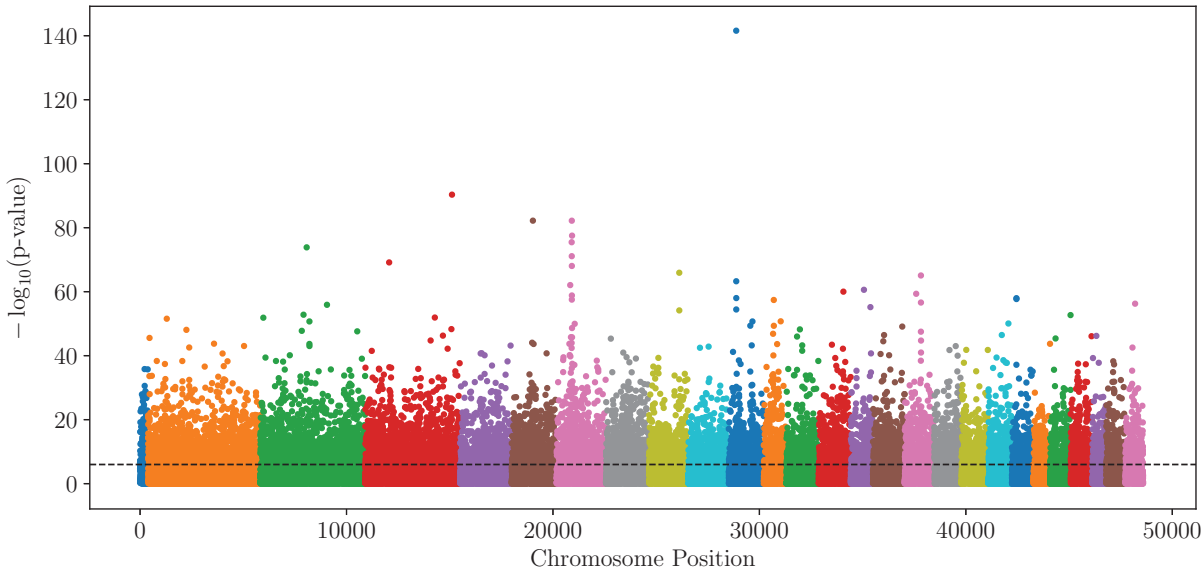
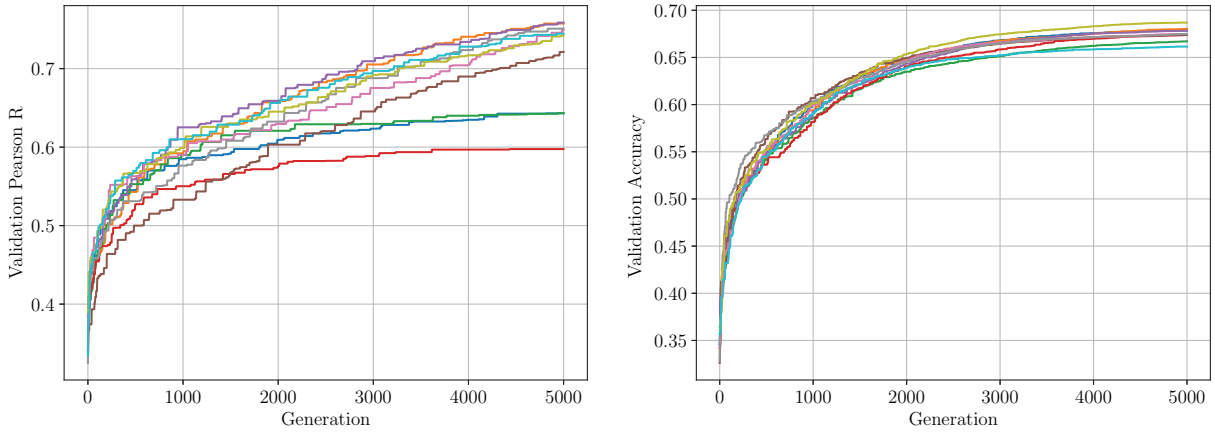
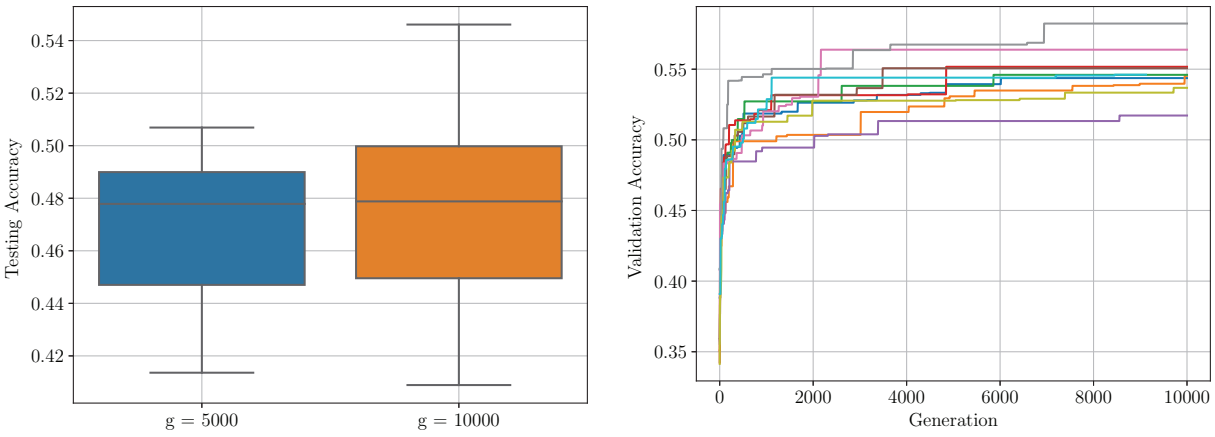


Figure A.1: Manhattan plot of the simulated data. The y-axis shows the $-\log_{10}$ of each effect's p-value. The x-axis describes the marker's location on the chromosome. Change in color denotes change in chromosome. The dotted line shows the Bonferonni threshold, i.e. $-\log_{10}(0.05/48588) \approx 5.9876$. Due to the simulation process and the fact that this simple plot does not account for population structure, most markers appear to be significant.



(a) (b)

Figure A.2: (a) Fixed subset intergenerational cross-validation convergence plot. (b) Fixed subset intragenerational cross-validation convergence plot.



(a) (b)

Figure A.3: (a) Boxplots comparing the testing error of the fixed subset Monte Carlo cross-validation experiment with 5,000 and 10,000 generations. The difference between the two is not significant, with $p = 0.2990$, though there is a higher maximum testing accuracy of 0.5350. (b) Convergence plot of the fixed subset Monte Carlo cross-validation experiment with 10,000 generations. Compare to Figure 4.3b.

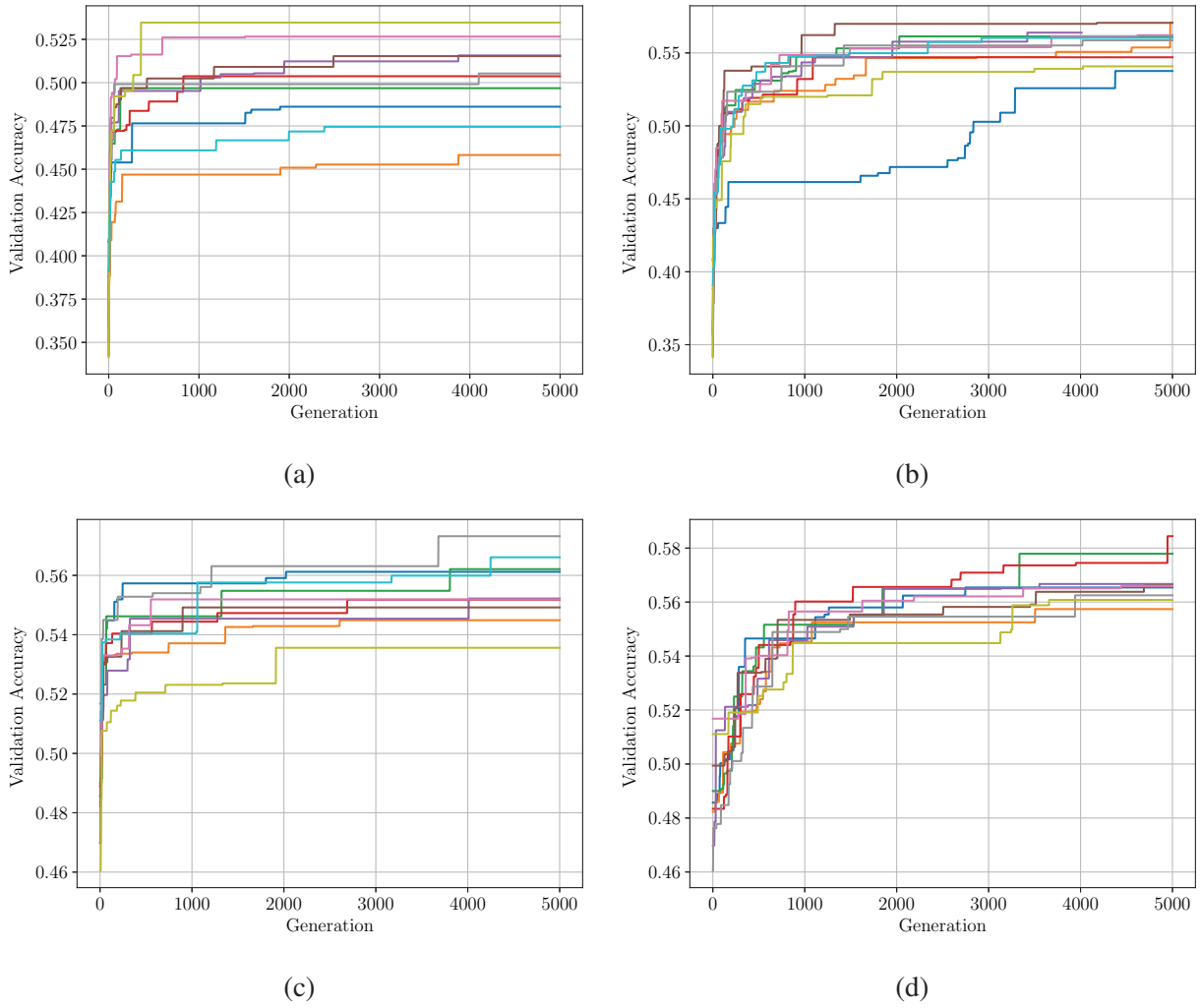
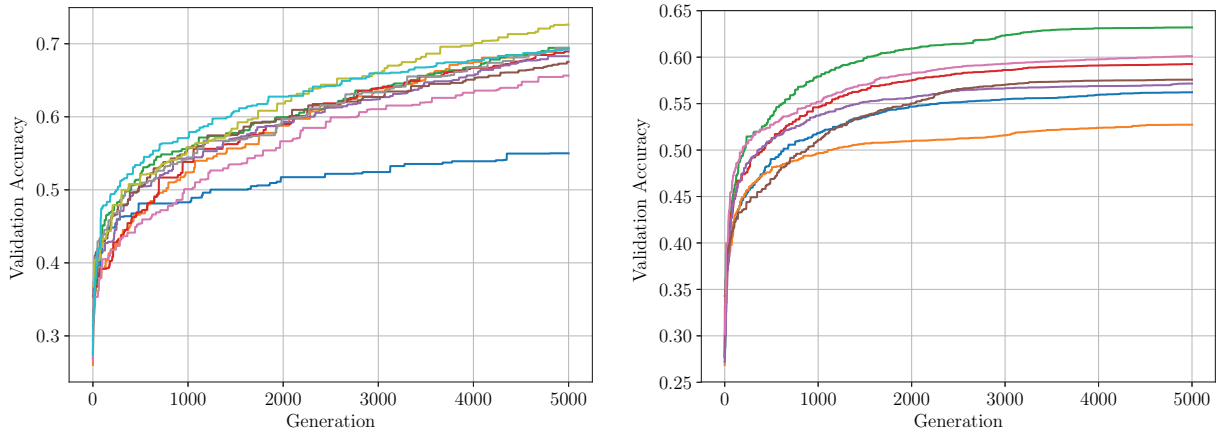


Figure A.4: (a) Convergence plot of fixed subset MDE_pBX with Monte Carlo cross-validation. (b) Convergence plot of fixed subset SaDE with Monte Carlo cross-validation. (c) Convergence plot of fixed subset seeded MDE_pBX with Monte Carlo cross-validation. (d) Convergence plot of fixed subset seeded SaDE with Monte Carlo cross-validation. Note that none of the plots exceed the heritability threshold $h \approx 0.6325$.



(a) (b)

Figure A.5: (a) Convergence plot of intergenerational cross-validation in the coevolution setting. (b) Convergence plot of intragenerational cross-validation in the coevolution setting. Missing replicates are due to memory overflow in both cases.

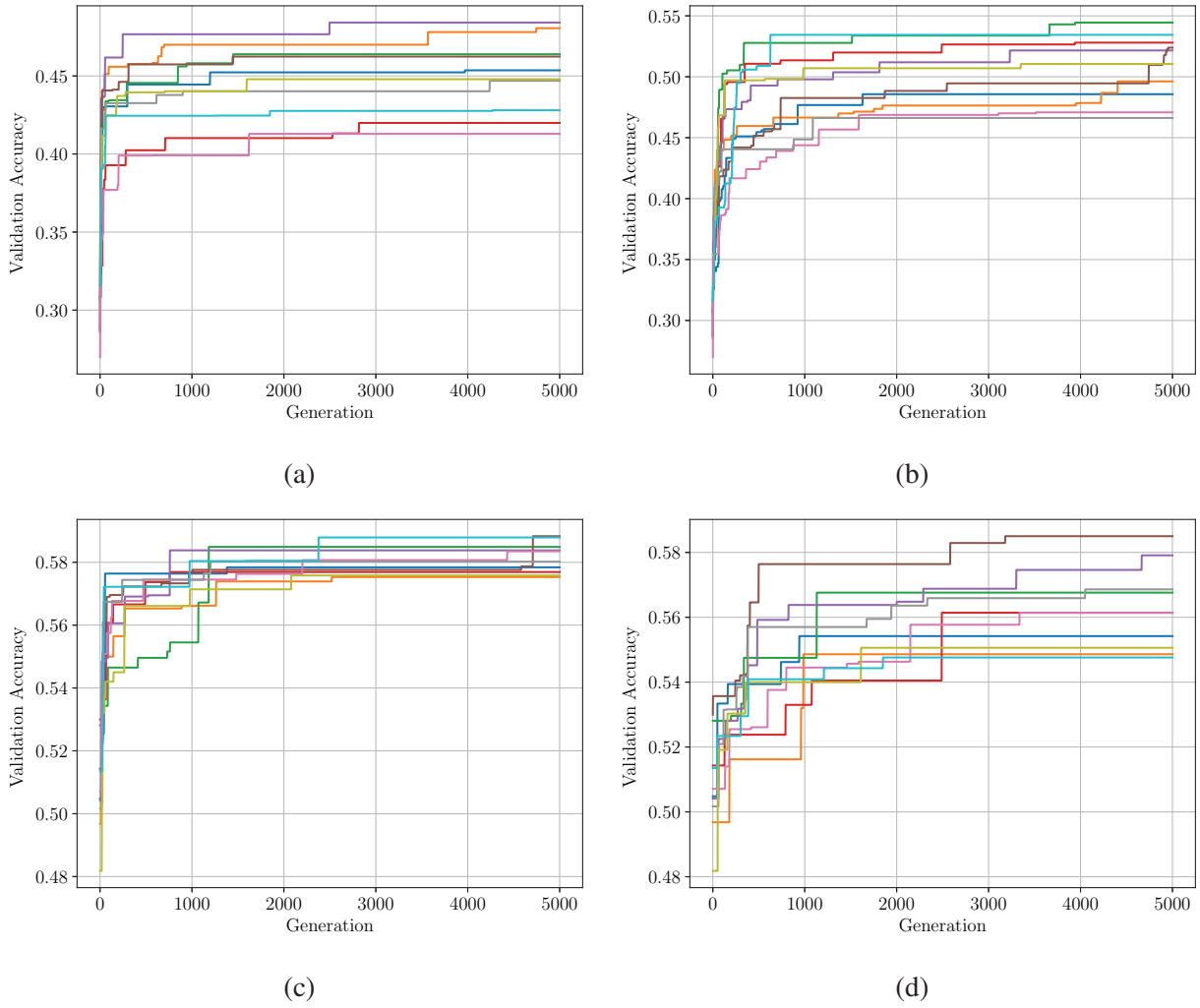


Figure A.6: (a) Convergence plot of coevolutionary MDE_pBX with Monte Carlo cross-validation. (b) Convergence plot of coevolutionary SaDE with Monte Carlo cross-validation. (c) Convergence plot of coevolutionary seeded MDE_pBX with Monte Carlo cross-validation. (d) Convergence plot of coevolutionary seeded SaDE with Monte Carlo cross-validation. Note that none of the plots exceed the heritability threshold $h \approx 0.6325$.

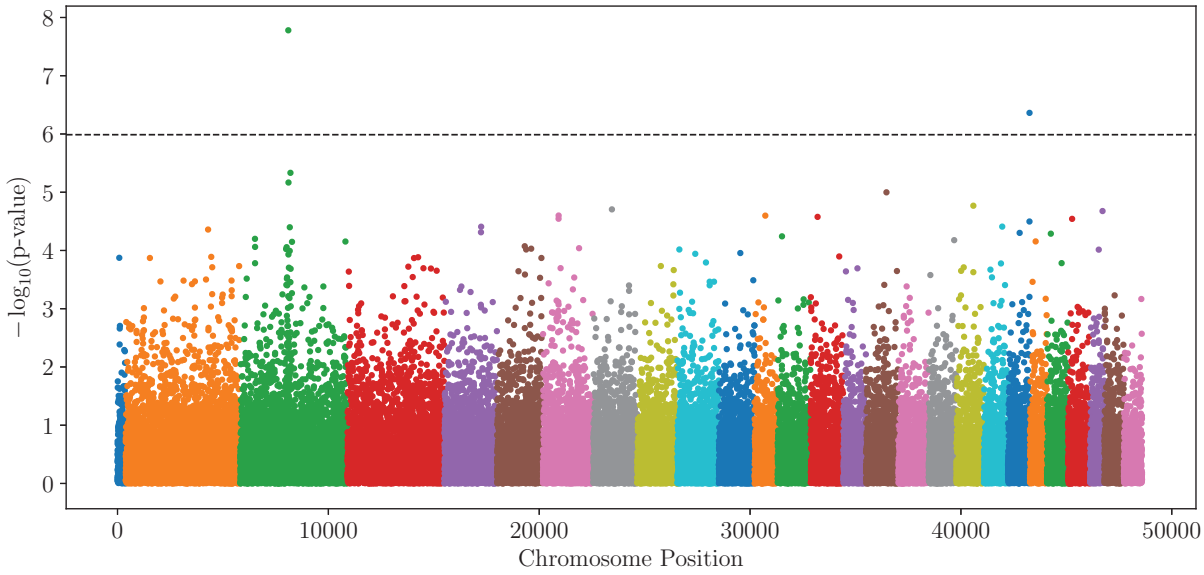


Figure A.7: Manhattan plot of the WEC data. The y-axis shows the $-\log_{10}$ of each effect's p-value. The x-axis describes the marker's location on the chromosome. Change in color denotes change in chromosome. The dotted line shows the Bonferonni threshold, i.e. $-\log_{10}(0.05/48588) \approx 5.9876$.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] T. H. E. Meuwissen, B. J. Hayes, and M. E. Goddard, "Prediction of total genetic value using genome-wide dense marker maps," *Genetics*, vol. 157, no. 4, pp. 1819–1829, 2001.

[2] M. E. Goddard and B. J. Hayes, "Genomic selection," *Journal of Animal Breeding and Genetics*, vol. 124, no. 6, pp. 323–330, 2007.

[3] R. Dalton, "No bull: genes for better milk," doi:10.1038/457369a, 2009.

[4] E. L. Heffner, M. E. Sorrells, and J.-L. Jannink, "Genomic selection for crop improvement," *Crop Science*, vol. 49, pp. 1–12, 2009, 1.

[5] E. L. Heffner, J. Jannink, and M. E. Sorrells, "Genomic selection accuracy using multifamily prediction models in a wheat breeding program," *The Plant Genome*, vol. 4, no. 1, pp. 65–75, 2011.

[6] Q. B. Kwong, A. L. Ong, C. K. Teh, F. T. Chew, M. Tammi, S. Mayes, H. Kulaveerasingam, S. H. Yeoh, J. A. Harikrishna, and D. R. Appleton, "Genomic selection in commercial perennial crops: Applicability and improvement in oil palm (*Elaeis guineensis* Jacq.)," *Scientific Reports*, vol. 7, no. 2872, 2017.

[7] G. Abraham, J. A. Tye-Din, O. G. Bhalala, A. Kowalczyk, J. Zobel, and M. Inouye, "Accurate and robust genomic prediction of celiac disease using statistical learning," *PLoS Genetics*, 2014.

[8] A. C. J. Janssens and C. M. van Duijn, "Genome-based prediction of common diseases: advances and prospects," *Human Molecular Genetics*, vol. 17, no. R2, pp. R166–R173, 2008.

[9] B. Hayes, P. Bowman, A. Chamberlain, and M. Goddard, "Invited review: Genomic selection in dairy cattle: Progress and challenges," *Journal of Dairy Science*, vol. 92, no. 2, pp. 433 – 443, 2009.

[10] B. J. Hayes, H. A. Lewin, and M. E. Goddard, "The future of livestock breeding: genomic selection for efficiency, reduced emissions intensity, and adaptation," *Trends in Genetics*, vol. 29, no. 4, pp. 206 – 214, 2013.

[11] R. R. Hofmann, "Evolutionary steps of ecophysiological adaptation and diversification of ruminants: a comparative view of their digestive system," *Oecologia*, vol. 78, no. 4, pp. 443–457, Mar 1989.

[12] B. J. Hayes, J. Pryce, A. J. Chamberlain, P. J. Bowman, and M. E. Goddard, "Genetic architecture of complex traits and accuracy of genomic prediction: coat colour, milk-fat percentage, and type in holstein cattle as contrasting model traits," *PLoS Genetics*, vol. 6, 2010.

[13] M. Goddard, B. Hayes, and T. Meuwissen, "Using the genomic relationship matrix to predict the accuracy of genomic selection," *Journal of Animal Breeding and Genetics*, vol. 128, no. 6, pp. 409–421, 2011.

[14] S. A. Clark, J. M. Hickey, and J. H. van der Werf, "Different models of genetic variation and their effect on genomic evaluation," *Genetics Selection Evolution*, vol. 43, no. 1, p. 18, May 2011.

[15] Y. C. J. Wientjes, R. F. Veerkamp, and M. P. L. Calus, "The effect of linkage disequilibrium and family relationships on the reliability of genomic prediction," *Genetics*, vol. 193, no. 2, pp. 621–631, 2013.

[16] N. Moghaddar, A. A. Swan, and J. H. J. Van Der Werf, "Comparing genomic prediction accuracy from purebred, crossbred and combined purebred and crossbred reference populations in sheep," *Genetics Selection Evolution*, vol. 46, no. 58, 2014.

[17] M. Goddard, "Genomic selection: prediction of accuracy and maximisation of long term response," *Genetica*, vol. 136, no. 2, pp. 245–257, Jun 2009.

[18] D. A. van Heel, L. Franke, K. A. Hunt, R. Gwilliam, A. Zhernakova, M. Inouye, M. C. Wapenaar, M. C. N. M. Barnardo, G. Bethel, G. K. T. Holmes, C. Feighery, D. Jewell, D. Kelleher, P. Kumar, S. Travis, J. R. Walters, D. S. Sanders, P. Howdle, J. Swift, R. J. Playford, W. M. McLaren, M. L. Mearin, C. J. Mulder, R. McManus, R. McGinnis, L. R. Cardon, P. Deloukas, and C. Wijmenga, "A genome-wide association study for celiac disease identifies risk variants in the region harboring il2 and il21," *Nature Genetics*, vol. 39, pp. 827–829, Jun 2007.

[19] P. M. Visscher, N. R. Wray, Q. Zhang, P. Sklar, M. I. McCarthy, M. A. Brown, and J. Yang, "10 years of GWAS discovery: Biology, function, and translation," *The American Journal of Human Genetics*, vol. 101, no. 1, pp. 5 – 22, 2017.

[20] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[21] C. Gondro, *Primer to Analysis of Genomic Data Using R*. Springer International Publishing, 2015.

[22] The 1000 Genomes Project Consortium, "A map of human genome variation from population-scale sequencing," *Nature*, vol. 467, pp. 1061–1073, Oct 2010.

[23] L. C. Kottyan, B. P. Davis, J. D. Sherrill, K. Liu, M. Rochman, K. Kaufman, M. T. Weirauch, S. Vaughn, S. Lazaro, A. M. Rupert, M. Kohram, E. M. Stucke, K. A. Kemme, A. Magnusen, H. He, P. Dexheimer, M. Chehade, R. A. Wood, R. D. Pesek, B. P. Vickery, D. M. Fleischer, R. Lindbad, H. A. Sampson, V. A. Mukkada, P. E. Putnam, J. P. Abonia, L. J. Martin, J. B. Harley, and M. E. Rothenberg, "Genome-wide association analysis of eosinophilic esophagitis provides insight into the tissue specificity of this allergic disease," *Nature Genetics*, vol. 46, pp. 895–900, Jul 2014.

[24] S.-H. Lee, B.-H. Park, A. Sharma, C.-G. Dang, S.-S. Lee, T.-J. Choi, Y.-H. Choy, H.-C. Kim, K.-J. Jeon, S.-D. Kim, S.-H. Yeon, S.-B. Park, and H.-S. Kang, "Hanwoo cattle: origin, domestication, breeding strategies and genomic selection," *Journal of Animal Science and Technology*, vol. 56, no. 1, p. 2, May 2014.

[25] W. S. Bush and J. H. Moore, "Chapter 11: Genome-wide association studies," *PLoS Computational Biology*, vol. 8, no. 12, December 2012.

[26] R. C. Johnson, G. W. Nelson, J. L. Troyer, J. A. Lautenberger, B. D. Kessing, C. A. Winkler, and S. J. O'Brien, "Accounting for multiple comparisons in a genome-wide association study (GWAS)," *BMC Genomics*, vol. 11, pp. 724–724, Dec 2010.

[27] B. S. F. Müller, L. G. Neves, J. E. de Almeida Filho, M. F. R. Resende, P. R. Muñoz, P. E. T. dos Santos, E. P. Filho, M. Kirst, and D. Grattapaglia, "Genomic prediction in contrast to a genome-wide association study in explaining heritable variation of complex growth traits in breeding populations of eucalyptus," *BMC Genomics*, vol. 18, p. 524, Jul 2017.

[28] C. Darwin, *On the Origin of Species by Means of Natural Selection*. London: Murray, 1859.

[29] C. Henderson, "Use of all relatives in intraherd prediction of breeding values and producing abilities," *Journal of Dairy Science*, vol. 58, no. 12, pp. 1910 – 1916, 1975.

[30] S. Ripatti, E. Tikkanen, M. Orho-Melander, A. S. Havulinna, K. Silander, A. Sharma, C. Guiducci, M. Perola, A. Jula, J. Sinisalo, M.-L. Lokki, M. S. Nieminen, O. Melander, V. Salomaa, L. Peltonen, and S. Kathiresan, "A multilocus genetic risk score for coronary heart disease: case-control and prospective cohort analyses," *The Lancet*, vol. 376, no. 9750, pp. 1393–1400, Oct 2010.

[31] J. B. Meigs, P. Shrader, L. M. Sullivan, J. B. McAteer, C. S. Fox, J. Dupuis, A. K. Manning, J. C. Florez, P. W. Wilson, R. B. D'Agostino Sr *et al.*, "Genotype score in addition to common risk factors for prediction of type 2 diabetes," *New England Journal of Medicine*, vol. 359, no. 21, pp. 2208–2219, 2008.

[32] M. R. Cooperberg, E. Davicioni, A. Crisan, R. B. Jenkins, M. Ghadessi, and R. J. Karnes, "Combined value of validated clinical and genomic risk stratification tools for predicting prostate cancer mortality in a high-risk prostatectomy cohort," *European Urology*, vol. 67, no. 2, pp. 326 – 333, 2015.

[33] M. Ayalew, H. Le-Niculescu, D. F. Levey, N. Jain, B. Changala, S. D. Patel, E. Winiger, A. Breier, A. Shekhar, R. Amdur, D. Koller, J. I. Nurnberger, A. Corvin, M. Geyer, M. T. Tsuang, D. Salomon, N. J. Schork, A. H. Fanous, M. C. O'Donovan, and A. B. Niculescu, "Convergent functional genomics of schizophrenia: from comprehensive understanding to genetic risk prediction," *Molecular Psychiatry*, vol. 17, p. 887, May 2012.

[34] G. Abraham and M. Inouye, "Genomic risk prediction of complex human disease and its clinical application," *Current Opinion in Genetics & Development*, vol. 33, pp. 10 – 16, 2015.

[35] L. Lello, S. G. Avery, L. Tellier, A. I. Vazquez, G. de los Campos, and S. D. H. Hsu, "Accurate genomic prediction of human height," *Genetics*, 2018.

[36] D. Habier, R. L. Fernando, and J. C. M. Dekkers, "The impact of genetic relationship information on genome-assisted breeding values," *Genetics*, vol. 177, no. 4, pp. 2389–2397, 2007.

[37] J. C. Whittaker, R. Thompson, and M. C. Denham, "Marker-assisted selection using ridge regression," *Genetical Research*, vol. 75, no. 2, p. 249–252, 2000.

[38] P. M. VanRaden, "Efficient methods to compute genomic predictions," *Journal of Dairy Science*, vol. 91, no. 11, pp. 4414–4423, 2008.

[39] D. Habier, R. L. Fernando, and D. J. Garrick, "Genomic BLUP decoded: A look into the black box of genomic prediction," *Genetics*, vol. 194, no. 3, pp. 597–607, 2013.

[40] G. Su, R. Brøndum, P. Ma, B. Guldbrandtsen, G. Aamand, and M. Lund, "Comparison of genomic predictions using medium-density (~54,000) and high-density (~777,000) single nucleotide polymorphism marker panels in nordic holstein and red dairy cattle populations," *Journal of Dairy Science*, vol. 95, no. 8, pp. 4657 – 4665, 2012.

[41] J. Zapata-Valenzuela, R. W. Whetten, D. B. Neale, S. E. McKeand, and F. Isik, "Genomic estimated breeding values using genomic relationship matrices in a cloned population of loblolly pine," *G3: Genes, Genomes, Genetics*, 2013.

[42] K. G. Dodds, B. Auvray, S.-A. N. Newman, and J. C. McEwan, "Genomic breed prediction in New Zealand sheep," *BMC Genetics*, vol. 15, no. 1, p. 92, Sep 2014.

[43] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[44] D. Habier, R. L. Fernando, K. Kizilkaya, and D. J. Garrick, "Extension of the bayesian alphabet for genomic selection," *BMC Bioinformatics*, vol. 12, no. 1, p. 186, May 2011.

[45] G. De Los Campos, H. Naya, D. Gianola, J. Crossa, A. Legarra, E. Manfredi, K. Weigel, and J. M. Cotes, "Predicting quantitative traits with regression models for dense molecular markers and pedigree," *Genetics*, vol. 182, no. 1, pp. 375–385, 2009.

[46] M. Erbe, B. J. Hayes, L. K. Matukumalli, S. Goswami, P. J. Bowman, C. M. Reich, B. A. Mason, and M. E. Goddard, "Improving accuracy of genomic predictions within and between dairy cattle breeds with imputed high-density single nucleotide polymorphism panels," *Journal of Dairy Science*, vol. 95, no. 7, pp. 4114–4129, 1 2011.

[47] T. Wang, Y.-P. P. Chen, M. E. Goddard, T. H. Meuwissen, K. E. Kemper, and B. J. Hayes, "A computationally efficient algorithm for genomic prediction using a bayesian model," *Genetics Selection Evolution*, vol. 47, no. 1, p. 34, Apr 2015.

[48] B. Li, N. Zhang, Y.-G. Wang, A. W. George, A. Reverter, and Y. Li, "Genomic prediction of breeding values using a subset of SNPs identified by three machine learning methods," *Frontiers in Genetics*, vol. 9, p. 237, 2018.

[49] T. K. Ho, "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition*, ser. ICDAR '95, vol. 1. Washington, DC, USA: IEEE Computer Society, 1995, pp. 278–282.

[50] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

[51] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

[52] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.

[53] N. F. Grinberg, A. Lovatt, M. Hegarty, A. Lovatt, K. P. Skøt, R. Kelly, T. Blackmore, D. Thorogood, R. D. King, I. Armstead, W. Powell, and L. Skøt, "Implementation of genomic prediction in *Lolium perenne* (l.) breeding populations," *Frontiers in Plant Science*, vol. 7, p. 133, Feb 2016.

[54] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning-based sequence model," *Nature Methods*, vol. 12, pp. 931 – 934, Aug 2015.

[55] P. Bellot, G. de los Campos, and M. Pérez-Enciso, "Can deep learning improve genomic prediction of complex human traits?" *Genetics*, 2018.

[56] D. Quang and X. Xie, "DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences," *Nucleic Acids Research*, vol. 44, no. 11, p. e107, Jun 2016.

[57] H. D. Patterson and R. Thompson, "Recovery of inter-block information when block sizes are unequal," *Biometrika*, vol. 58, no. 3, pp. 545–554, 1971.

[58] N. S. Forneris, A. Legarra, Z. G. Vitezica, S. Tsuruta, I. Aguilar, I. Misztal, and R. J. C. Cantet, "Quality control of genotypes using heritability estimates of gene content at the marker," *Genetics*, vol. 199, no. 3, pp. 675–681, Mar 2015.

[59] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar 2003.

[60] J. Biesiada and W. Duch, "A Kolmogorov-Smirnov correlation-based filter for microarray data," in *Neural Information Processing*, M. Ishikawa, K. Doya, H. Miyamoto, and T. Yamakawa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 285–294.

[61] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[62] W. Altidor, T. M. Khoshgoftaar, and J. Van Hulse, "Robustness of filter-based feature ranking: A case study." in *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, 2011.

[63] V. Bolón-Canedo, N. Sánchez-Marono, A. Alonso-Betanzos, J. M. Benítez, and F. Herrera, "A review of microarray datasets and applied feature selection methods," *Information Sciences*, vol. 282, pp. 111–135, 2014.

[64] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 02, pp. 185–205, 2005.

[65] H. D. Pereira, J. M. Soriano Viana, A. C. B. Andrade, F. Fonseca e Silva, and G. P. Paes, "Relevance of genetic relationship in gwas and genomic prediction," *Journal of Applied Genetics*, vol. 59, no. 1, pp. 1–8, Feb 2018.

[66] W. Zhou, C. Zhou, G. Liu, and H. Zhu, "Feature selection for microarray data analysis using mutual information and rough set theory," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2006, pp. 492–499.

[67] M. Robnik-Šikonja and I. Kononenko, "An adaptation of relief for attribute estimation in regression," in *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*, vol. 5, 1997, pp. 296–304.

[68] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 856–863.

[69] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 1–14, Jan 2013.

[70] F. Santosa and W. Symes, "Linear inversion of band-limited reflection seismograms," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1307–1330, 1986.

[71] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[72] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[74] L. Sheng, R. Pique-Regi, S. Asgharzadeh, and A. Ortega, "Microarray classification using block diagonal linear discriminant analysis with embedded feature selection," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 04 2009, pp. 1757–1760.

[75] D. F. Schwarz, I. R. König, and A. Ziegler, "On safari to random jungle: A fast implementation of random forests for high-dimensional data," *Bioinformatics*, vol. 26, no. 14, pp. 1752–1758, 2010.

[76] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in Bioinformatics*, vol. 2015, p. 198363, Jun 2015.

[77] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 389–422, Jan 2002.

[78] S. Maldonado, R. Weber, and J. Basak, "Simultaneous feature selection and classification using kernel-penalized support vector machines," *Information Sciences*, vol. 181, no. 1, pp. 115 – 128, 2011.

[79] N. E. Aboudi and L. Benhlima, "Review on wrapper feature selection approaches," in *2016 International Conference on Engineering MIS (ICEMIS)*, Sept 2016, pp. 1–5.

[80] A. Wayne Whitney, "A direct method of nonparametric measurement selection," *Computers, IEEE Transactions on*, vol. 20, pp. 1100 – 1103, 10 1971.

[81] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119 – 1125, 1994.

[82] C. Ambroise and G. J. McLachlan, "Selection bias in gene extraction on the basis of microarray gene-expression data," *Proceedings of the National Academy of Sciences*, vol. 99, no. 10, pp. 6562–6566, 2002.

[83] Q. Liu, A. H. Sung, Z. Chen, J. Liu, X. Huang, and Y. Deng, "Feature selection and classification of MAQC-II breast cancer and multiple myeloma microarray gene expression data," *PLoS One*, vol. 4, no. 12, pp. 1–24, 12 2009.

[84] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Nov 1995, pp. 1942–1948.

[85] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2, pp. 95–99, Oct 1988.

[86] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[87] A. Huning, *ARSP: Archiv für Rechts- und Sozialphilosophie / Archives for Philosophy of Law and Social Philosophy*, vol. 62, no. 2, pp. 298–300, 1976.

[88] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[89] R. Luque-Baena, D. Urda, M. G. Claros, L. Franco, and J. Jerez, "Robust gene signatures from microarray data using genetic algorithms enriched with biological pathway keywords," *Journal of Biomedical Informatics*, vol. 49, pp. 32 – 44, 2014.

[90] R. M. Luque-Baena, D. Urda, J. L. Subirats, L. Franco, and J. M. Jerez, "Application of genetic algorithms and constructive neural networks for the analysis of microarray cancer data," *Theoretical Biology and Medical Modelling*, vol. 11, no. Suppl 1, pp. S7–S7, May 2014.

[91] M. Perez and T. Marwala, "Microarray data feature selection using hybrid genetic algorithm simulated annealing," in *2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel*, Nov 2012, pp. 1–5.

[92] E. K. Tang, P. N. Suganthan, and X. Yao, "Feature selection for microarray data using least squares svm and particle swarm optimization," in *2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, Nov 2005, pp. 1–8.

[93] S. Li, X. Wu, and M. Tan, "Gene selection using hybrid particle swarm optimization and genetic algorithm," *Soft Computing*, vol. 12, no. 11, pp. 1039–1048, Sep 2008.

[94] L. Rastrigin, *Systems of extremal control*. Moscow: Nauka, 1974, p. 632.

[95] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Computing*, vol. 17, no. 6, pp. 619 – 632, 1991.

[96] A. P. Piotrowski, "Review of differential evolution population size," *Swarm and Evolutionary Computation*, vol. 32, pp. 1 – 24, 2017.

[97] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, Oct 2009.

[98] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *2005 IEEE Congress on Evolutionary Computation*, vol. 2, Sept 2005, pp. 1785–1791 Vol. 2.

[99] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 482–500, April 2012.

[100] R. D. Al-Dabbagh, F. Neri, N. Idris, and M. S. Baba, "Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy," *Swarm and Evolutionary Computation*, 2018.

[101] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[102] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y.-p. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *Natural Computing*, pp. 341–357, Jan 2005.

[103] S. Das and P. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Jadavpur University, Tech. Rep., Sept 2018.

[104] W. S. Sakr, R. A. EL-Sehiemy, and A. M. Azmy, "Adaptive differential evolution algorithm for efficient reactive power management," *Applied Soft Computing*, vol. 53, pp. 336 – 351, 2017.

[105] W. Kwedlo, "A clustering method combining differential evolution with the k-means algorithm," *Pattern Recognition Letters*, vol. 32, no. 12, pp. 1613 – 1621, 2011.

[106] D. K. Tasoulis, V. P. Plagianakos, and M. N. Vrahatis, "Differential evolution algorithms for finding predictive gene subsets in microarray data," in *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, K. Karpouzis, and M. Bramer, Eds. Boston, MA: Springer US, 2006, pp. 484–491.

[107] H. A. Al-Mamum, P. Kwan, S. Clark, S. H. Lee, K. D. Song, S. H. Lee, and C. Gondro, "Genomic best linear unbiased prediction using differential evolution," in *Proceedings of the AAABG 21st Conference*, 2015, pp. 145–148.

[108] C. Esquivelzeta-Rabell, H. A. Al-Mamum, S. H. Lee, K. D. Song, and C. Gondro, "Evolving to the best SNP panel for hanwoo breed proportion estimates," in *Proceedings of the AAABG 21st Conference*, 2015, pp. 473–476.

[109] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.

[110] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, no. 1, pp. 61–75, 1956.

[111] R. N. Khushaba, A. Al-Ani, and A. Al-Jumaily, "Differential evolution based feature subset selection," in *2008 19th International Conference on Pattern Recognition*, Dec 2008, pp. 1–4.

[112] A. C. Nearchou and S. L. Omirou, "Differential evolution for sequencing and scheduling optimization," *Journal of Heuristics*, vol. 12, no. 6, pp. 395–411, Dec 2006.

[113] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, no. 1, pp. 50–60, 03 1947.

[114] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–.

[115] STUDENT, "The probable error of a mean," *Biometrika*, vol. 6, no. 1, pp. 1–25, 1908.

[116] A. Xavier, S. Xu, W. Muir, and K. Rainey, "NAM: Association studies in multiple populations," *Bioinformatics*, vol. 31, no. 23, pp. 3862–3864, 2015.