BOOSTING THE PERFORMANCE OF MULTI-OBJECTIVE EPISTASIS DETECTION

Francisco Gonçalves
Instituto Superior Técnico

In recent years, studies have shown that genetic interaction can play a major role in the occurrence of diseases that affect human beings. An increasing number of approaches to detect these interactions have been developed, but due to the complexity of certain diseases, higher orders of genetic interaction need to be studied. Due to the size of the search space, fast and efficient approaches are necessary to identify high-order interactions. To tackle this problem, this Thesis proposes improvements over the multi-objective genetic algorithm, NSGA-II, and presents two parallelization designs of the proposed algorithm. Interaction orders up to 10 were tested, with three different problem instances, on an Intel Xeon Gold multicore multiprocessor system. Great improvements over all interaction orders was achieved, reaching up to 196×. In the parallel designs, great scalability was obtained in higher interaction orders in one of the approaches. Overall a combined speedup, over the base work, of more than 2000× was achieved.

Keywords
Epistasis detection, Parallelism, SNP detection, Genetic Algorithm

I. Introduction

Human genetics, the study of inheritance in human beings, has as one of its goals the identification of genes that increase the risk of a certain disease. Identifying those genes can lead to the development of new treatment, prevention and diagnostic methods, thus leading to the improvement of human health. Even though the great importance of this task, it is not easily accomplished due to the complex interaction between multiple genes and multiple environmental factors [1].

The focus of this work is on single nucleotide polymorphisms (SNPs), and their complex interactions. A SNP is a single point in a DNA sequence that differs between individuals [2]. The traditional approach to linking the state of these SNPs to the risk of a certain disease, is to measure the genotypes of people with, case samples, and without, control samples, a certain disease with a relevant number of SNPs, these studies are called Genome-Wide Association Studies (GWAS). Afterwards each SNP is individually tested for a direct association with the disease under study. This approach has had some success, but by only examining the association of single SNPs, these studies ignore complex interactions that may be critical to understanding more complex diseases [2], [3]. The combined effect of multiple interacting SNPs is known as epistasis [4]. Epistasis seems to be the main responsible for the appearance of complex traits like human diseases [5]. Methods designed for single SNP detection do not work for these interactions, which cannot be modeled [6].

A number of approaches have been proposed to detect epistasis, these can generally be classified into three categories: exhaustive search, stochastic search and machine-learning approaches [7]. Since a number of studies revealed that these single correlation model or objective function approaches perform inconsistently with different disease models, and that even the same approach will often vary when applied to different disease models, epistasis detection researches have been moving towards multi-objective methods, methods with multiple decision criteria [3]. These methods have been described as having better accuracy and thus avoiding those inconsistencies [3]. Epistasis introduces a greater level of complexity, since the number of SNPs involved in the interactions (k) can vary from 2 to unknown numbers. The search space of this optimization problem grows exponentially with the growth of the epistasis interaction order [8]. More specifically, for a k-order epistasis and M SNPs, the number of possible combinations is given by $\frac{M!}{k!\times(M-k)!}$. Additionally, to increase the reliability of a study a greater number of case-control samples is needed. Due to the complexity of epistasis interaction detection brute-force approaches are impractical [9]. Recently, genetic and evolutionary algorithms have been used to solve high-order epistasis interactions, of up to k=8 [9].

Epistasis detection raises computational challenges since analyzing every SNP combination is not viable at a genome-wide scale. Using as a reference the works from [9] and [3], this problem can be surpassed. Even with genetic algorithms, the complexity of the problem grows linearly with the number of individuals in the studied population, but it has an exponential growth with the increase of the interaction order degree. Thus high order epistasis detection is at the moment an unpractical problem to tackle.

This thesis has the Non-dominated Sorting Genetic Algorithm (NSGA-II) of [9] as base, with the main goal being reducing the time taken in epistasis detection, in order to reach epistasis interaction order of 10, previously not tackled. To achieve this goal, improvements over the main functions of NSGA-II are proposed. Additionally, to further improve computational times and to exploit CPU potential, based on the work of [10], two parallelization schemes using OpenMP are proposed. In this context parallelization is not trivial due to an intrinsic serial component of the algorithm and data dependencies but this combination of approaches has already proven results in other complex optimization problems [11].

In the first section (Section I), an introduction to the work of this report has been done. Thereafter, this works is divided in the following way:

- In Section II, a summary regarding state-of-the-art is presented, the problem tackled in this work is formulated, and an explanation of NSGA-II, which serves as base for this work, is made.

- In Section IV, proposed methods to improve execution time in epistasis detection are presented, and in order to tackle high order epistasis interactions two parallel implementations are introduced.
- In Section IV, experimental results of the proposed improvement methods are presented, experimental results from the two proposed parallel designs are presented and analyzed, a comparison of execution times with NSGA-II is presented, and a brief evaluation of solution quality is made.
- In Section VIII, the conclusions obtained from the work performed in this Thesis are presented, as well as, possible future works.

II. Related Work

Results from Genome-Wide Association Studies (GWAS) have been focused on single associations between the disease under study and Single Nucleotide Polymorphisms (SNPs), where only addictive effects are considered. Limited results have been achieved with these studies [12], due to these single associations only explaining a small fraction of the estimated heritability [13] (heritability is the proportion of variation in a trait explained by inherited genetic variants). One factor that could explain this, is the multiple interaction between SNPs, known as epistasis [14]. Recently a great number of statistical methods have been developed for epistasis detection in GWAS, ranging from exhaustive search to various machine learning approaches.

The search space for epistasis detection grows exponentially with the interaction order. This leads to exhaustive search algorithms being limited to an order of interaction of two or three SNPs, due to the huge computational time required to finish the analysis [8]. In order to reduce execution time, works using Graphical Processing Units (GPUs) like [15] can be found. [16] proposes a combination of a GPU and a FPGA, to solve exhaustive third order epistasis interaction detection, claimed to be faster than proposals using GPU only, as [17]. Finally, [18] proposes using FPGAs to tackle third order epistasis detection, but stating that problems with half a million SNPs could take as much as three years. All the previous works can only tackle second and third order epistasis interactions, leading to the conclusion that for higher order of interaction non-exhaustive approaches are needed.

To efficiently tackle higher than two epistasis interaction orders, an exhaustive exploration of the full search space can not be executed. Following this idea, [19] tries to identify variants that have a non-zero interaction effect with other variants, in this way avoiding the direct search for pairwise or higher order interactions. In [20], Sure Independence Screening (SIS) is used to reduce the search space, with SIS the SNPs are ordered according to their marginal correlation with the trait and afterwards a number of best candidate SNPs is selected to be tested. All the previous works with an incorrect filtering could be losing important SNPs.

Another approach to this problem are evolutionary algorithms, the approach tackled in this thesis. Evolutionary algorithms are a more recent trend in epistasis detection. MACOED, is a memory-based multi-objective ant colony optimization algorithm, which is able to retain non-dominated solutions found in past iterations, to solve the space and time complexity for high-dimension problems, for pairwise interactions, with proven results [3]. [21] combines a Bayesian scoring function with an evolutionary-based heuristic search algorithm to solve epistasis interaction. FSHA-SED is a multiobjective second order epistasis detection method based on the harmony search algorithm [22]. Finally the Non-dominated sorting algorithm of [23] was applied to epistasis detection in [9], reducing the memory required for epistasis detection and tackling high interaction orders of up to 8 SNPs.

Every epistasis detection algorithm requires a metric to evaluate each solution, i.e., an objective score function. Traditionally single objective scores have been used in epistasis detection methods, but more recently a trend of multiobjective, mostly two, scores has emerged due to some inconsistencies in single objective methods and a better performance using multiobjective approaches [3]. A multiobjective problem is aimed at finding a set of multiple solutions, called Pareto optimal solutions or Pareto front. The Pareto front represents the trade-off between the two objective score functions. Some of the most used scores are the Akaike information criterion (AIC) [3], [9] and Bayesian network based scores [3], [9], [22], [21]. Other scores like Gini-score can be found [22].

III. Problem Formulation

The combined effect of multiple interacting single-nucleotide polymorphisms (SNPs), a single point in a DNA sequence that differs between individuals, is known as epistasis. Epistasis seems to be the main responsible for the appearance of complex traits like human diseases [5]. Genome-Wide Association Studies (GWAS) provide SNP data from case samples, samples with the disease under study, and control samples, samples known not to have the disease under study, that can be used to determine epistasis interactions. The goal of this work is to identify which interacting SNP combinations have a higher probability of being responsible for the manifestation of a certain disease in the case samples, by analysing the SNP values present in the genotype of the dataset samples.

To make those analysis it is necessary to process a dataset containing N case-control samples, M SNPs and the information of the disease state, i.e. if the sample is a case or a control. For every sample the file contains the genotype value of every SNP marker under study. Each SNP is represented by a number from 0 to 2, to codify the possible allele representations, i.e. to codify each variant form of a given gene, 0 for homozygous major allele, 1 for heterozygous allele, or 2 for homozygous minor allele. The disease state is represented by either 0, a control sample, or 1, a case sample. After analysis, the output (for a epistasis interaction order k) corresponds to a list of, k SNP markers.

Epistasis detection is a complex problem due to the number of SNPs and the epistasis interaction order, when the interaction order increases the search space increases exponentially.

The number of possible combinations for a k-order epistasis analysis and M SNP markers is given by: $\frac{M!}{k!(M-k)!}$.

IV. Description of the methods

This work has as base the structure of the state-of-the-art multi-objective tools for epistasis detection. More specifically the multi objective evolutionary search engine, Non-dominated Sorting Genetic Algorithm II (NSGA-II), a bi-objective optimization problem with the widely used K2 and AIC scores, based on the work of [23]. This algorithm has as its objective the improvement of a population of candidate solutions to form a Pareto front. In this implementation an individual (or solution) is represented by an integer array of k-elements, where each element represents an SNP from the input dataset. In this way, this array codifies the interactive effect between the specified k SNPs.

NSGA-II can be generally described in 7 steps. Step 1: Creation of a random population to initialize the search for the Pareto front. Step 2: Non dominated sorting process to rank the initial population. Step 3: Generation of the first offspring population based on the evolutionary operators. Step 4: Rank the combined population, parent and offspring populations. Step 5: Based on the determined ranks and crowding distance of each individual, determine the population of the next generation. Step 6: Generation of a new population. Steps 4 to 6 are repeated until a certain number of generation is reached, this number of generations can vary according to the problem tackled. Step 7: Final rank of the population, of which the first rank corresponds to the Pareto front approximation of the problem.

In this bi-objective optimization problem, the first objective score function adopted is built upon the concept of a Bayesian network, a statistical model that uses a directed acyclic graph to represent a set of random variables and their conditional dependencies [24], [25]. In this kind of graph, the association between an SNP x and a certain disease state y is represented by a direct edge linking from node x to y. Thus, the K2 score can be expressed as:

$$K2 = \sum_{i=1}^{I} \left(\sum_{b=1}^{r_i+1} log(b) - \sum_{j=1}^{J} \sum_{d=1}^{r_{ij}} log(d) \right), \tag{1}$$

where I refers to the number of possible genotype combinations (for an epistasis interaction order k, $I = 3^k$ as an SNP can show a value of 0, 1, or 2), J the number of disease states (J = 2 in case-control scenarios), r_i represents the frequency of the i-th genotype combination in the samples at the SNP nodes $[x_1, x_2, ..., x_k]$, i.e. the number of times each combination of the analyzed SNPs appears in the samples, and r_{ij} the number of samples where the disease node takes the j-th state and the SNPs take the i-th genotype combination, i.e. the number of of times each SNP combination appears for each of the disease states of the samples. The goal in this problem is to minimize this function, the lower the score value the stronger the association between the SNP set and the disease.

The second objective score function is based on a logistic regression, which has been widely used in state of the art multiobjective tools [26], that calculates the likelihood of occurrence of the disease for a certain SNP combination under a multilocus interaction model. The Akaike information criterion (AIC) is a metric that reflects the model fitness to the dataset and the complexity of the model used [27]. In this model $log(lik)$ represents the maximized log-likelihood of the model, and d the number of free parameters.

$$AIC = -2log(lik) + 2d \tag{2}$$

To compute the likelihood an additive-interactive model from North et al. (2005)[28] was used according to Equation 3.

$$lik = log\frac{p}{1-p} = \mu + \sum_{i=1}^{k} \beta_i x_i + \xi \prod_{i=1}^{k} x_i \tag{3}$$

In the previous equation x_i represents the i-th SNP in the evaluated solution, μ is a mean term of population prevalence and sample sizes, β and ξ refer to additive and interactive effect factors respectively, and p represents the conditional probability $P(y = 1|[x_1, ..., x_k])$. As in K2 score, this is also an objective function score to minimize.

V. Proposed methods for reducing epistasis detection time

In this section the proposed optimizations to improve the state-of-the-art methods, in order to undertake high order epistasis study and to accelerate the study of low order epistasis interactions, will be explained. In order to fully exploit the capabilities of the CPU and to further reduce the time taken in epistasis detection, the potential parallelism will be exploited. Two independent approaches will be investigated, under the work of [10]. The two approaches were implemented using the standard OpenMP.

A. *Generation of the population*

Upon the generation of the offspring population, the computation of the objective scores is performed immediately for every individual. With these objective score values, it is possible to check if an individual will not be in the parent population of the next generation. If an individual is dominated by the last front in the parent population, it will have at least N individuals in the ranks above, and thus will not make the cut of N individuals chosen for the next parent population. By checking the quality of each solution upon generation it is possible to reduce the number of offspring individuals introduced in the offspring population. As a result, having a smaller offspring population, the time spent sorting, determining ranks and assigning a crowding distance can be reduced.

Since a more diverse population produces better results and the best individuals remain in the first rank of the population, it is not necessary to check the same individuals again, i.e. generating an individual more than once. For this purpose, it is necessary to keep a database of individuals generated and evaluated. A solution for this problem that is both time and memory efficient is a hash table. Each entry of this hash table has the identifiers x_i of the SNPs markers that compose that particular solution. In a hash table, the time necessary to find a solution can be reduced up to Hash$_{Size}$ times. A simple and fast dispersion function is used: (SNP1+ ... + SNPN)/Hash$_{Size}$.

B. Sorting of the population

In a genetic algorithm, the selection of the best individuals to proceed to the next generation is essential. To determine the Pareto ranks of the solutions in the population, NSGA-II relies on the use of fast non-dominated sorting, after which the candidates with the lowest ranks are selected, while the best solutions within the same rank are determined according to the crowding distance[23]. The ranking algorithm proposed in this Thesis focuses on the necessary conditions for the solutions to belong to the same rank. Assuming a minimization context, two solutions Υ_1 and Υ_2 necessarily belong to the same rank if and only if their scores for both objective functions are equal, i.e., $K2(\Upsilon_1) = K2(\Upsilon_2)$ and $AIC(\Upsilon_1) = AIC(\Upsilon_2)$ (equality condition, $\Upsilon_1 \approx \Upsilon_2$) or if they improve each other in different objective functions, i.e., $K2(\Upsilon_1) < K2(\Upsilon_2)$ ($\Upsilon_1 \xrightarrow{\text{K2}} \Upsilon_2$) and $AIC(\Upsilon_2) < AIC(\Upsilon_1)$ ($\Upsilon_2 \xrightarrow{\text{AIC}} \Upsilon_1$). Furthermore, any given solution Υ_3 will belong to the same rank as the solutions Υ_1 and Υ_2 if and only if:

- $\Upsilon_3 \approx \Upsilon_1$ or $\Upsilon_3 \approx \Upsilon_2$ (equality conditions); or
- $\Upsilon_3 \xrightarrow{\text{K2}} \Upsilon_1 \xrightarrow{\text{K2}} \Upsilon_2$ and $\Upsilon_2 \xrightarrow{\text{AIC}} \Upsilon_1 \xrightarrow{\text{AIC}} \Upsilon_3$; or
- $\Upsilon_1 \xrightarrow{\text{K2}} \Upsilon_3 \xrightarrow{\text{K2}} \Upsilon_2$ and $\Upsilon_2 \xrightarrow{\text{AIC}} \Upsilon_3 \xrightarrow{\text{AIC}} \Upsilon_1$; or
- $\Upsilon_1 \xrightarrow{\text{K2}} \Upsilon_2 \xrightarrow{\text{K2}} \Upsilon_3$ and $\Upsilon_3 \xrightarrow{\text{AIC}} \Upsilon_2 \xrightarrow{\text{AIC}} \Upsilon_1$.

A set of solutions Υ_i will belong to the same rank if and only if the equality conditions are satisfied or if the order of solutions remains the same when they are sorted in the increasing order of their K2 score and in the decreasing order of their AIC score, i.e., $\Upsilon_1 \xrightarrow{\text{K2}} \Upsilon_2 \xrightarrow{\text{K2}} ... \xrightarrow{\text{K2}} \Upsilon_n$ and $\Upsilon_1 \xleftarrow{\text{AIC}} \Upsilon_2 \xleftarrow{\text{AIC}} ... \xleftarrow{\text{AIC}} \Upsilon_n$. By relying on this observation, the proposed ranking procedure relies on sorting the candidate solutions in a increasing order of their K2 score, with the aim of reassuring the condition $\Upsilon_1 \xrightarrow{\text{K2}} \Upsilon_2 \xrightarrow{\text{K2}} ... \xrightarrow{\text{K2}} \Upsilon_n$. Figure 1a provides an example of a set of sorted solutions, which are enumerated according to their increasing K2 score value. After sorting, two adjacent solutions Υ_i and Υ_{i+1} will belong to the same rank if and only if:

- $\Upsilon_i \approx \Upsilon_{i+1}$ (#1.1); or
- $\Upsilon_{i+1} \xrightarrow{\text{AIC}} \Upsilon_i$ (#1.2), since the condition $\Upsilon_i \xrightarrow{\text{K2}} \Upsilon_{i+1}$ is necessary satisfied due to the sorting.

In all other cases, two adjacent solutions Υ_i and Υ_{i+1} will belong to different ranks, most notably if:

- $K2(\Upsilon_i) = K2(\Upsilon_{i+1})$, the solution with a lower AIC score will be placed in the lower rank (#2.1); or
- $\Upsilon_i \xrightarrow{\text{AIC}} \Upsilon_{i+1} (\Upsilon_i \xrightarrow{\text{K2}} \Upsilon_{i+1})$, the solution Υ_i will belong to the lower rank (#2.2); or
- $AIC(\Upsilon_i) = AIC(\Upsilon_{i+1}) (\Upsilon_i \xrightarrow{\text{K2}} \Upsilon_{i+1})$,the solution Υ_i will belong to the lower rank (#2.3).

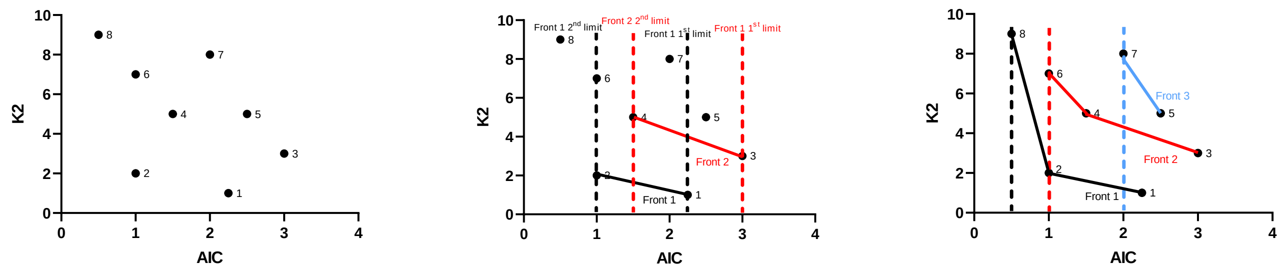
In this ranking procedure, the creation of ranks starts from the solution with the minimum K2 score. This process is depicted in Figure 1b when creating the first two fronts (ranks). Since the solution 1 has the minimum K2 score, it is placed in the Front 1. Solution 2 will join solution 1 in Front 1 since it satisfies #1.2 (i.e., it has a lower AIC score than the solution 1). The solution 3 cannot belong to Front 1 due to #2.2 (both K2 and AIC scores are higher when compared to the solutions 1 and 2), thus it is placed in the Front 2. Since solution 4 satisfies #1.2 it is placed in the Front 2 together with the solution 3. Solution 5 will belong to Front 3 due to #2.1 (equal K2 cores and a higher AIC score when compared to the solution 4 from Front 2). In this ranking procedure, each front (rank) is additionally assigned with the minimum AIC score among the solutions in that rank. For example, for the fronts depicted in Figure 1b, the AIC score of solution 2 will be considered for Front 1, while the AIC score of solution 4 will be assigned to Front 2 (see the vertical dashed lines). The consideration of the minimum AIC scores for each front is crucial to allow insertion of additional solutions in the already created fronts. Given the fact that all other non-examined solutions (that do not satisfy #2.1) must have a higher K2 score than the already ranked solutions (due to the initial sorting), the solution under evaluation can only belong to an existing front if and only if its AIC score is strictly lower than the AIC scores of all solutions already included in that front (see condition #1.2). As a result, before creating a new front, the AIC score of the currently evaluated solution is compared with the minimum AIC score of the existing fronts (starting from the last created front up to Front 1). If there exists a front with a minimum AIC score that is higher than the AIC score of the current solution, the solution is appended to that front and the minimum AIC score of the front is updated with the AIC score of the appended solution. This process is depicted in Figure 1c. Once all solutions are ranked, the best popSize solutions are selected (starting from Front 1). The individuals belonging to the same front are selected according to the crowding distance. It is worth noting that, in the proposed approach, the calculation of crowding distance does not require any additional sorting, since the applied K2 sorting and the solution appending process guarantee that the solutions within the same rank are sorted according to their increasing K2 score and decreasing AIC score. As such, since the first and the last solutions mark the relative points for crowding distance calculation, the crowding distance of the solutions is directly obtained by iterating over the list of individuals of every front.

Algorithm 1 presents the pseudo-code of the proposed sorting and ranking algorithm. In this algorithm, p represents the solution under evaluation, F_i the list of solutions belonging to Front i, $N[i]$ the minimum value in AIC score for Front i, $rank$ represents the highest rank assigned (the worst rank), and x represents the rank that will be assigned to solution p.

C. Objective Score Functions

In order to evaluate the quality of each solution objective scoring functions are needed. Each solution needs to be evaluated in each of the objective scores, with N solutions being generated at every generation efficient objective scoring functions are of the upmost importance. In the following sub-sections proposed improvements over both objective score functions will be explained.

As it can be observed in Equation 1, there are two main stages to be performed when computing Bayesian K2 score. The first is the determination of frequencies for each genotype combination i (at the SNPs of the evaluated solution) with respect to the j-th disease state r_{ij} , as well as the



(a) Example sorted by K2 (b) Step 4 of ranking (c) Final result

Figure 1: Example of the proposed Ranking Algorithm

Algorithm 1 Sorting and Ranking

```
 1: $Pop \leftarrow Sort_{K2}(Pop)$
 2: $p \leftarrow Pop[1]$
 3: $p_{rank} = 0$
 4: $F_1 \leftarrow p$
 5: $N[1] \leftarrow p_{AIC}$
 6: for $i = 2$ to $PopSize$ do
 7:     $x = rank$
 8:     $p \leftarrow Pop[i]$
 9:     while $p_{AIC} < N[x]$ do
10:         $x = x - 1$
11:     end while
12:     if $x = rank$ then
13:         $rank = rank + 1$
14:     end if
15:     $p_{rank} = x$
16:     $F_x \leftarrow p$
17:     $N[x] = p_{AIC}$
18:     $p \leftarrow CrowdDistance$
19: end for
```

overall observed frequency r$_i$. The second stage refers to the calculation of K2 score as a sum of the logarithms with respect to the determined r$_i$ and r$_{ij}$ frequencies. The first and more computationally expensive step implies traversing through the values of each SNP marker of each sample. Instead of computing the index of the array, that stores the frequency of the i-th genotype combination in the samples at the SNP nodes using the formula

$$Index = \sum_{j=0}^{k} SNP_j \times 3^{k-1-j}. \tag{4}$$

To efficiently determine frequency of each genotype combination, a fast way of indexing an array based on the values of the SNP markers for each sample is necessary. The index of genotype combination ($observedGen$), of the array r, can be obtained by recursively applying the expression $observedGen = 3 \times observedGen + D[s, x]$, where different genotype values $D[s, x]$ (observed for the s-th sample at the SNP x) are used as the relative indexing offsets. This process is illustrated in Figure 2, for the interaction order k = 3 and the genotype values D[s] = 1,2,1, whose genotype combination index in the r array ($observedGen$) corresponds to the value of 16. Depending on the disease state either X (for controls) or Y (for cases) will be incremented. Once the genotype combination index is determined, the respective frequency position (r_{ij}, $i = observedGen$) is incremented depending on the j-th disease state.

As for the second stage of the objective score K2, it is necessary to perform 3^k partial K2 score calculations, as it can be seen in Equation 1. To avoid repeated computations in the evaluation of an individual and in the evaluation of different individuals, a look-up table is used. Since the upper limit of this frequency is the sample size, it is possible to create a look-up table with a reasonable size. By pre-computing the logarithm values and storing them, at initialization, in the table, those expensive computations can be replaced by a simple read from an array. Thus, the most efficient approach to determine K2 score is compute according to Equation 1, and reading the necessary values from the proposed look up table.

The computation of AIC score represents the main bottleneck in terms of execution time. This is due to the heavy computations to estimate the likelihood.

To determine the likelihood of occurrence of the disease for the SNPs in the evaluated solution, an iterative method is used. According to Equation 3, this procedure implies solving the system of equations $P = XB$, where P is the vector of probabilities $log(\frac{p_s}{1-p_s})$ calculated for each sample s, X a design matrix with rows $[1, x_1, x_2, ..., x_k, x_1x_2...x_k]$ and x_i being the i-th SNP marker of a given sample in the evaluated solution, and B the parameters of the model $[\mu, \beta_1, \beta_2, ..., \beta_k, \xi]$. The parameters B must be estimated prior to these calculations, by applying the iterative re-weighted least squares (IRLS) method [29]:

$$B_{g+1} = (X^T W_g X)^{-1} X^T (W_g X B_g + y - \lambda_g), \tag{5}$$

where g is the current iteration of the estimation method, $\lambda_g = \frac{e^{XB_g}}{1+e^{XB_g}}$ represents the expected values, y $\in \{0, 1\}$ the response variables (the observed disease state), and $W = diag(\lambda_g(1-\lambda_g))$ a diagonal weighting matrix. The parameters of the model are iteratively calculated based on this procedure, whose stop criterion is set to an improvement precision of 0.001 according to the literature [3].

Computing $X^T W X$ is quite expensive since it depends on the sample size, as does the estimation of the parameters λ,

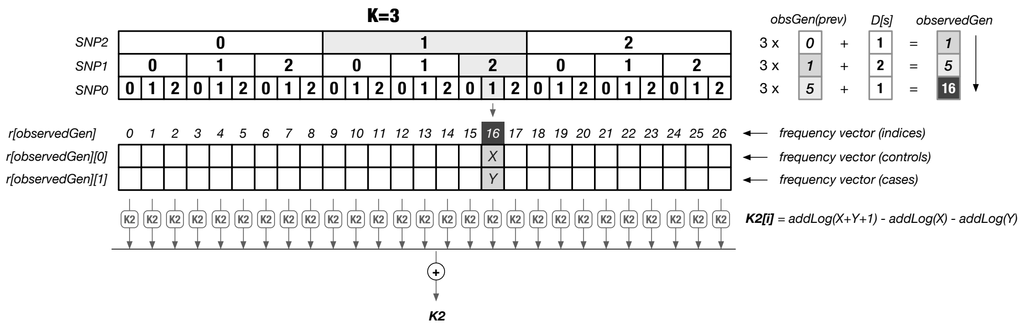


Figure 2: Example, for epistasis size 3, of index calculation

y, and consequently W. It can be noted that the maximum number of genotype combinations depends on the epistasis interaction order, more specifically it is 3^k. When $3^k < SampleSize$ there have to be samples with the same genotype combination, i.e. there are samples with the same SNP values in the respective SNP markers. For example, for k=2, there are only 9 possible combinations of the values of the SNP markers to be evaluated. Since the average number of samples in an input file is large, this opens the possibility to improve the way X^TWX and the parameters are computed. Instead of solving the system of equations for every sample and repeating computations, it is possible to solve it for every genotype combination and weighting its importance by multiplying by its frequency. This way, it is possible to reduce the number of computations when the number of samples is greater than 3^k.

D. Parallelization of the improved method

In order to improve the computational times of the algorithm two parallelization approaches are investigated, under the work of [10].

The first approach considered is an iteration-level strategy that allows the definition of a problem-independent approach to parallelize epistasis detection methods, without modifying the behaviour of the search engine compared to the serial version. The generation and processing of new candidate solutions is tackled by distributing different solutions to different execution threads. Each thread is responsible for the generation, comparison with the database and scoring a certain number of new solutions. This parallelization is achieved by using worksharing directives like `#pragma omp for`, while serial components of this application are ensured by using the clause `#pragma omp single`. Potential data dependencies are protected using *omp locks*. Algorithm 2 presents the pseudo-code of the proposed approach.

The second approach follows a trend of parallelizing at the solution level, by parallelizing the computations performed at each individual solution. The potential parallelism of this trend depends on the specific variables and implementations of the optimization problem. The goal is to reduce time in the problem dependent computations since those are the most expensive computations. More specifically, in the proposed approach, the two objective functions, thus parallel designs of both scores are proposed.

Algorithm 2 Problem-independent parallel design

```
 1: Initialize Population (Population)
 2: #pragma omp parallel
 3: while ! stop criterion (maximum generations) do
 4:     #pragma omp single
 5:     Fronts ← Ranking and Crowding (Population)
 6:     Parents ← Parent Identification (Fronts) /*selecting
        best popSize individuals*/
 7:     #pragma omp for schedule (dynamic)
 8:     for i = 1 to popSize do
 9:         p1, p2 ← Parent Selection (Parents)
10:         q ← Crossover and Mutation (p1, p2)
11:         q ← Validate Offspring Solution (q, HashTable)
12:         q.K2, q.AIC ← Evaluate Offspring Solution (q)
13:         Population ← Integrate Offspring Solution (q)
14:     end for
15: end while
16: Return Pareto Solutions
```

Algorithm 3 presents the pseudo-code of the parallelization of the Bayesian K2 score, which consists of two stages. The first stage refers to a parallelization of the calculation of the genotype frequencies. The second stage corresponds to the final calculation of K2 score, expressed in Equation 1.

As for the second objective score, the pseudo-code for its parallelization can be seen in Algorithm 4.

VI. Experimental Results and Analysis

In this section the results of the improvements will be discussed, their viability and corresponding time improvement.

For experimentation purposes, three problem instances with different characteristics (in terms of numbers of SNPs and case/control samples) have been considered:

- DB23x10000: a real-world breast cancer dataset composed of 10,000 samples (5,000 cases and 5,000 controls), each one characterized by 23 SNPs from the genes COMT, CYP19A1, ESR1, PGR, SHBG, and STS [30];
- DB1000x4000: a benchmark dataset containing 4,000 samples (2,000 controls and 2,000 cases) with 1,000 SNPs, generated by using the GAMETES software [31];
- DB31341x146: a benchmark dataset containing 146 samples (50 controls and 96 cases) with 31,341 SNPs, generated by using the GAMETES software [31].

Algorithm 3 Problem-dependent parallel design - K2 score

```
1: $r \leftarrow$ Initialize Genotype Frequencies /* $r = 0$ */
2: /* Identifying frequencies for each potential genotype combination and disease state */
3: #pragma omp for schedule (guided)
4: for s=1 to N do
5:     for i=k to 1 do
6:         observedGen $\leftarrow$ Identify Genotype Combination (observedGen, dataset[s][q.SNP[i]])
7:     end for
8:     D_State $\leftarrow$ Get Disease State(dataset[s][M+1])
9:     #pragma omp atomic
10:    r[observedGen][D_State]++
11: end for
12: /* Applying Equation 1 with look-up table */
13: #pragma omp for reduction(+:K2)
14: for i=1 to $3^k$ do
15:     K2 $\leftarrow$ K2_Comp(r[i][0], r[i][1], r[i][0]+r[i][1])
16: end for
17: return K2
```

Algorithm 4 Problem-dependent parallel design - AIC score

```
1: $B \leftarrow$ Initialize Model Parameters /* $B = 0$*/
2: #pragma omp for schedule (guided)
3: for s=1 to N do
4:     X $\leftarrow$ Initialize Matrix (dataset[s][q.SNP[i]]) /* $\forall$ i,i = 1 to k */
5: end for
6: while ! stop criterion ($B$.improvement < 0.001) do
7:     /* Computing $B_r = X^T(W_i X B_i + y - \lambda_i)$*/
8:     #pragma omp for schedule (guided)
9:     for s=1 to N do do
10:        $\lambda \leftarrow$ Compute Expected Values ($X[s], B$)
11:        $y \leftarrow$ Get Disease State (dataset[s][M+1])
12:        $W \leftarrow ComputeWeights(\lambda)$
13:        $B_r \leftarrow ComposeB_r entry(W, X[s], B, y, \lambda)$
14:    end for
15:    /* $ComputingB_l = X^T W X$*/
16:    #pragma omp for schedule (guided)
17:    for s=1 to N do
18:        $B_l \leftarrow ComposeB_l entry(W, X[s])$
19:    end for
20:    $B \leftarrow B_l B_r$
21: end while
22: #pragma omp for reduction(+:$AIC$)
23: for s=1 to N do
24:     $lik \leftarrow$ Compute likelihood ($X[s], B$)
25:     $AIC \leftarrow$ Compute AIC Score ($lik$)
26: end for
27: $AIC = AIC + 2d$
28: return $AIC$
```

DB31341x146				
	4 cores		32 cores	
Dim_Epi	SU	EF	SU	EF
k=2	3.12	77.92%	7.60	23.73%
k=4	3.59	89.71%	16.51	51.59%
k=6	3.86	96.46%	24.64	77.01%
k=8	3.77	94.35%	24.67	77.10%
k=10	3.96	98.94%	28.66	89.56%
DB23x10000				
	4 cores		32 cores	
Dim_Epi	SU	EF	SU	EF
k=2	2.74	68.54%	4.69	14.66%
k=4	3.85	96.30%	21.38	66.80%
k=6	3.90	97.58%	26.30	82.19%
k=8	3.91	97.87%	29.21	91.29%
k=10	3.94	98.61%	31.21	97.54%
DB1000x4000				
	4 cores		32 cores	
Dim_Epi	SU	EF	SU	EF
k=2	3.72	92.96%	17.76	55.50%
k=4	3.68	92.03%	21.19	66.21%
k=6	3.85	96.25%	21.49	67.15%
k=8	3.88	97.09%	25.53	79.79%
k=10	3.85	96.30%	23.96	74.88%

Table I: Speedup and efficiency values for independent parallelization approach

The input parameters were configured according to the state of the art literature[3], [9]. The crossover probability was set to 50%, the mutation probability to 20%, and the mutation range to 30%. The population size and the stop criterion were established to 64 individuals and 100 generations for DB23x10000, 100 individuals and 1,500 generations for DB1000x4000, and 500 individuals and 2,000 generations for DB31341x146, due to the differences in the search space between each file, a different number of generations and a different population size is needed to reach the optimal solution in each dataset.

All experimental results were obtained by averaging at least five independent runs on a multicore multiprocessor system composed of two Intel Xeon Gold 6140 at 2.3GHz (a total of 36 physical cores in the system) with 25M Cache and 4x16GB DDR4-2666 RAM. CentOS 7.5 is used as operating system and the software tested in this research was compiled by using GCC 4.8.5. All experimental times presented are in seconds.

VII. Evaluation of Parallel Performance

In the following two sections the performance and efficiency of both parallel designs, and a comparison between both designs and their scalability, will be presented and evaluated. A comparison between parallelization schedules will be presented. Additionally a comparison with SOA method NSGA-II, showing the total improved time, will be made.

A. Problem-independent design

Table I presents the mean speedups and efficiencies of the independent parallelization design approach VII-A, for 4 and 32 cores. Across all datasets effective reductions in time were obtained, with efficiencies reaching 97.5% high order epistasis interactions (k=10).

When implementing a parallelization one thing to consider is the loop scheduling, i.e. the assignment of iterations in a parallelized loop to each execution thread. When using *Openmp*

DB31341x146				
	4 cores SU		32 cores SU	
Dim_Epi	Dynamic	Guided	Dynamic	Guided
k=2	3.12	3.32	7.60	10.30
k=4	3.59	3.63	16.51	19.68
k=6	3.86	3.66	24.64	24.86
k=8	3.77	3.86	24.67	24.23
k=10	3.96	3.996	28.66	27.74
DB23x10000				
	4 cores SU		32 cores SU	
Dim_Epi	Dynamic	Guided	Dynamic	Guided
k=2	2.74	2.65	4.69	4.92
k=4	3.85	3.75	21.38	21.61
k=6	3.90	3.89	26.30	26.31
k=8	3.91	3.90	29.21	29.27
k=10	3.94	3.95	31.21	31.18
DB1000x4000				
	4 cores SU		32 cores SU	
Dim_Epi	Dynamic	Guided	Dynamic	Guided
k=2	3.72	3.93	17.76	19.45
k=4	3.68	3.80	21.19	18.89
k=6	3.85	3.77	21.49	21.63
k=8	3.88	3.83	25.53	25.26
k=10	3.85	3.87	23.96	24.25

Table II: Speedup values for schedule dynamic and guided used in the independent parallelization approach

DB31341x146				
	4 cores		32 cores	
Dim_Epi	SU	EF	SU	EF
k=2	0.92	23.01%	0.39	1.23%
k=4	1.28	32.07%	0.58	1.81%
k=6	1.48	37.00%	0.67	2.08%
k=8	1.46	36.55%	0.85	2.65%
k=10	1.27	31.69%	1.35	4.22%
DB23x10000				
	4 cores		32 cores	
Dim_Epi	SU	EF	SU	EF
k=2	1.53	38.24%	4.82	15.05%
k=4	2.11	52.71%	7.93	24.78%
k=6	2.55	63.69%	11.47	35.83%
k=8	2.64	65.90%	13.24	41.39%
k=10	2.93	73.25%	13.87	43.34%
DB1000x4000				
	4 cores		32 cores	
Dim_Epi	SU	EF	SU	EF
k=2	1.45	36.22%	3.23	10.10%
k=4	2.05	51.24%	5.82	18.19%
k=6	2.35	58.65%	7.91	24.72%
k=8	2.70	67.39%	8.94	27.93%
k=10	2.80	69.91%	9.23	28.85%

Table III: Speedup and efficiency values for dependent parallelization approach

different schedules are available. Between all those schedules the most efficient schedules are the dynamic schedules due to the random nature of the problem, so the most effective schedules the dynamic and guided, thus their respective speedups are presented in Table II. Dynamic schedule was ran with a chunk size of one. As it can be seen the differences are not considerable, except for DB31341x146, which has a higher number of iterations to parallelize, due to a higher population in each generation, and an AIC score computationally less heavy due to a smaller sample size. In this dataset for the lower epistasis interaction orders, of k=2 and 4, overheads in thread management, for the dynamic schedule, can be noted, due to the small chunk size of one. Otherwise they are complimentary and picking one over the other is not straight forward.

B. Problem-dependent design

The second parallelization design is a parallelization at the solution level, i.e. at the objective score level, since those are the most time consuming functions of the total execution time. The potential parallelism of this approach is directly related to the number of case/control samples in the dataset and with the epistasis interaction order under study (k). Table III presents the average speedups and efficiencies for the experiments ran on this problem-dependent design, for 4 and 32 cores, across all three datasets. Since this parallelization is at a much lower level, lower speedups, and consequently efficiencies, were expected due to a higher number of computational constraints, data dependencies and synchronization needs.

Figures 3, 4 and 5 present the comparison between the parallel-dependent and parallel-independent approaches, for the files DB31431x146, DB23x10000 and DB4000x1000, respectively. Each figure shows the results for 4, 8, 16 and 32 cores, for epistasis interaction size of 10.

A problem-dependent parallelization is not very effective on an input file with the characteristics of DB31341x146, due to the small sample size, of just 146 samples. The input file DB23x10000 represents a file with a low number of SNP markers and a high number of samples. This represents a relatively small search space but objective score functions that take longer to compute. Thus contributing for the best results in both parallelization approaches of the three input files. The third input file represents a more balanced problem but on a higher number of SNP markers. The results for this file represent a reasonably good scalability for the parallel independent design, even though this file having a load imbalance. As for the dependent design the scalability is poor since it reaches its upper limit with 16/32 cores.

The limiting factor for a parallel dependent design is data consistency. Since it is a parallelization at a lower level it is necessary to ensure a consistent access to the variables in memory. With the increase of the number of cores this becomes a bottleneck. In the objective score function K2, where the computations are simpler and more related to memory reads and writes, for a higher number of cores engaged, an increase in the total time spent in that function is seen.

C. Total performance evaluation

Due to the main goal of this work being the improvement of execution time of the state-of-the-art multiobjective tools, Table IV presents a comparison between the execution times of the SOA method NSGA-II with a serial implementation of the proposed improvements and a 36 core parallelization of the proposed problem-independent design. The relative speedup, of both implementations, with NSGA-II are also presented, representing the total speedup obtained with this work.

For the proposed serial implementation these speedups range from 1,93 to 196. DB31341x146 presents the higher speedups due to the larger benefit from an improved database of solutions generated and tested. Across all three datasets lower epistasis interaction orders present larger speedups than

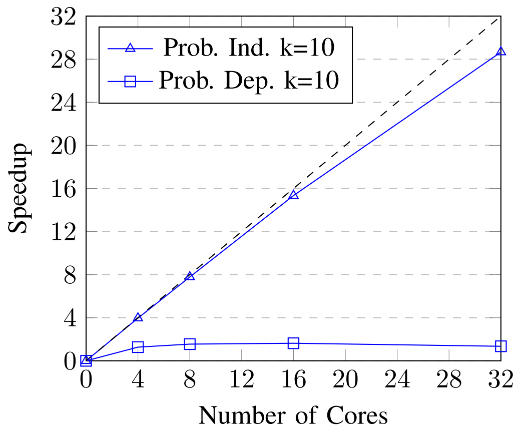


Figure 3: Speedup comparison, for DB31341x146, between problem-dependent and problem-independent approaches.

DB31341x146		
	k=2	k=10
NSGA-II	4596.83	17420.79
Serial	23.41	870.66
Speedup	196.37	20.01
36 cores	2.26	27.43
Speedup	2036.27	635.21
DB23x10000		
	k=2	k=10
NSGA-II	1.83	331.30
Serial	0.05	172.00
Speedup	33.55	1.93
36 cores	0.02	5.51
Speedup	112.94	60.17
DB1000x4000		
	k=2	k=10
NSGA-II	480.19	5922.10
Serial	32.74	1753.20
Speedup	16.24	3.38
36 cores	1.36	67.03
Speedup	354.11	88.35

Table IV: Execution times of NSGA-II vs serial implementation of the proposed improvements vs 36 cores parallelization of proposed problem-independent design

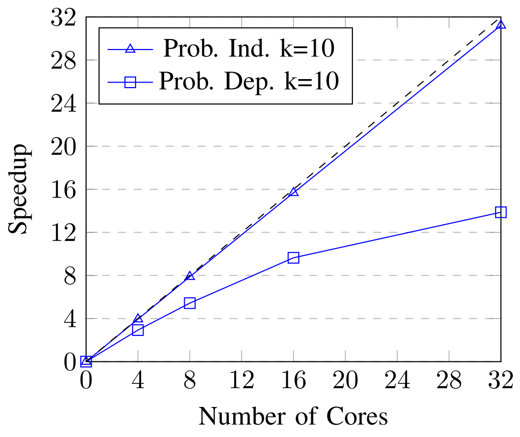


Figure 4: Speedup comparison, for DB23x10000, between problem-dependent and problem-independent approaches.

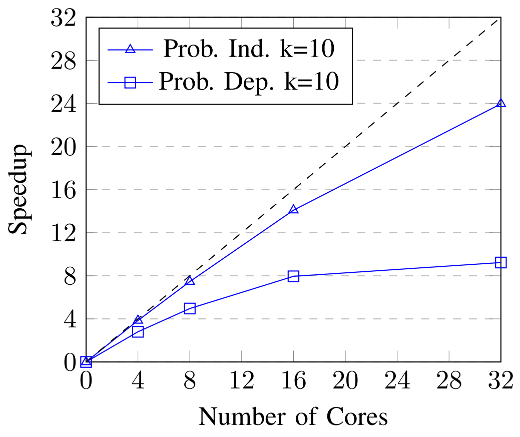


Figure 5: Speedup comparison, for DB4000x1000, between problem-dependent and problem-independent approaches.

high epistasis interaction orders due to improved AIC score computation method.

For the proposed parallelization design, the problem-independent approach, the speedups range from 60 to 2036. An average total speedup of 834 for k=2 and 261 for k=10 was obtained with the proposed methods. With this parallelization it was possible to achieve an average of 1.5 seconds in execution time for k=2 and 33.3 for k=10, presenting a reduction of several hours of computation to a few dozens of seconds in the worst case for high epistasis interaction orders.

VIII. Conclusion

Epistasis detection is a hard to tackle problem, with its complexity increasing exponentially with the increase of interaction order. With the growing number of GWAS, and consequently genetics research, epistasis detection has become an increasingly important problem to solve. Hence the importance of reducing execution times without losing solution quality.

To tackle this problem state-of-the-art multiobjective epistasis detection methods were revised and improved in high-order scenarios. Additionally two parallel implementations were introduced to exploit the opportunities of a CPU based method in a multicore multiprocessor system based on the latest generation of Intel Xeon CPU architectures. Real-world and benchmark problem instances have been used to evaluate the parallel performance and solution quality for epistasis interaction orders of k = 2, 4, 6, 8 and 10.

In terms of a serial implementation improvements of 196x where achieved. With the main contribution being a high reduction on the time taken in the computation of AIC score, with a theoretical limit of $\frac{Sample\_Size}{3^k}$. This means that the higher the sample size the higher the gains with this method, this opens a door to an increase in the size of the studies. Important gains were also achieved in the management of the database containing all the evaluated solutions, contributing to a possible increase in the number of SNP markers in the dataset without a decrease in performance. A new method of bi-objective sorting was also introduced with important relative gains over the previous fast non-dominant sort.

As for a parallel implementation, two designs were explored, a problem-independent and a problem-dependent. The problem-independent design proved to have the higher gains and scalability, reaching speedups of 31.21 with 32 cores, for epistasis interactions of k=10. For lower epistasis interaction order speedups of 17.76 were obtained, these speedups were not because of the improvements already made in the serial implementation, and thus it was a small problem to be paralellized. In absolute values the times obtained were in the order of just a few seconds. In terms of scheduling the two most efficient were dynamic and guided, the two being complimenting each other in different situations.

The second parallelization design, a problem-dependent approach, at the objective score level, revealed to be the less efficient and with poor scalability. Besides being harder to implement due to data consistency problem, in reproducing operations due to rounding differences in floating point numbers representation, it also has a difficult data consistency management. This leads to a parallel-dependent implementation being a less recommendable approach.

All the improvements were made without losing solution quality over the state-of-the-art methods NSGA-II and MACOED.

References

[1] Moore, J.H., Hahn, L.W., Ritchie, M.D., Thornton, T.A., White, B.C., "Application of genetic algorithms to the discovery of complex models for simulation studies in human genetics," *Proceedings of the Genetic and Evolutionary Computation Conference. Genetic and Evolutionary Computation Conference; 1150-1155. Epub Jul 1*, 2002.

[2] Greene, C. S., Himmelstein, D. S., Moore, J. H., "A model free method to generate human genetics datasets with complex gene-disease relationships," *Pizzuti C., Ritchie M.D., Giacobini M. (eds) Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. EvoBIO 2010. Lecture Notes in Computer Science, vol 6023. Springer, Berlin, Heidelberg*, 2010.

[3] Jing, P., Shen, H., "Macoed: a multi-objective ant colony optimization algorithm for snp epistasis detection in genome-wide association studies," *Bioinformatics 31(5):634–641*, 2015.

[4] Rieger, R., Michaelis, A., Green, A., "A glossary of genetics and cytogenetics," *Springer*, 1968.

[5] Mackay, T.F., "Epistasis and quantitative traits: using model organisms to study gene-gene interactions," *Nature Reviews Genetics 15(1): 22–33*, 2014.

[6] Moore, J.H. et al., "Bioinformatics challenges for genome-wide association studies," *Bioinformatics, 26, 445–455*, 2010.

[7] Shang, J., Zhang, J., Sun, Y., Liu, D., Ye, D., Yin, Y., "Performance analysis of novel methods for detecting epistasis," *BMC Bioinformatics 2011 12:475*, 2011.

[8] Ritchie, M.D., "Finding the epistasis needles in the genome-wide haystack," *Epistasis, Methods in Molecular Biology vol. 1253). Springer, pp. 19–33*, 2014.

[9] Gallego-Sánchez, D., Granado-Criado, J.M., Santander-Jiménez, S., Rubio-Largo, A., Vega-Rodríguez, M.A., "Parallel multi-objective optimization for high-order epistasis detection," *Algorithms and Architectures for Parallel Processing, LNCS, volume 10393. Springer International Publishing, pp. 523–532*, 2017.

[10] Talbi, E.G., "Parallel evolutionary combinatorial optimization," *Springer Handbook of Computational Intelligence. Springer, pp. 1107–1125*, 2015.

[11] Luna, F., Alba, E., "Parallel multiobjective evolutionary algorithms," *Springer Handbook of Computational Intelligence. Springer, pp. 1017–1031.*, 2015.

[12] Niel, C., Sinoquet, C., Dina, C. Rocheleau, G., "A survey about methods dedicated to epistasis detection," *Front Genet; 6: 285*, 2015.

[13] Manolio, T.A., Collins, F.S., Cox, N.J., Goldstein, D.B., Hindorff, L.A., Hunter, D.J., McCarthy, M.I., Ramos, E.M., Cardon, L.R., Chakravarti, A., Cho, J.H., Guttmacher, A.E., Kong, A., Kruglyak, L., Mardis, E., Rotimi, C.N., Slatkin, M., Valle, D., Whittemore, A.S., Boehnke, M., Clark, A.G., Eichler, E.E., Gibson, G., Haines, J.L., Mackay. T.F., McCarroll, S.A., Visscher, P.M., "Finding the missing heritability of complex diseases," *Nature*, 2009.

[14] Phillips, P. C., "Epistasis—the essential role of gene interactions in the structure and evolution of genetic systems," *Nat Rev Genet. 2008 Nov; 9(11): 855–867*, 2008.

[15] Wan, X., Yang, C., Yang, Q., Zhao, H., Yu, W., "The complete compositional epistasis detection in genome-wide association studies," *BMC Genetics 14 (1)*, 2013.

[16] Wienbrandt, L., Kässens, J. C., Hbenthal, M., Ellinghaus, D., "Fast genome-wide third-order snp interaction tests with information gain on a low-cost heterogeneous parallel fpga-gpu computing architecture," *Procedia Computer Science 108*, 2017.

[17] González-Domínguez, J., Schmidt, B., "Gpu-accelerated exhaustive search for third-order epistatic interactions in case–control studies," *Journal of Computational Science*, 2015.

[18] Kässens, J. C., Wienbrandt, L., González-Domínguez, J., Schmidt, B., Schimmler, M., "High-speed exhaustive 3-locus interaction epistasis analysis on fpgas," *Journal of Computational Science 9 131–136*, 2015.

[19] Crawford, L., Zeng, P., Mukherjee, S., Zhou, X., "Detecting epistasis with the marginal epistasis test in genetic mapping studies of quantitative traits," *PLOS Genetics 13*, 2017.

[20] Mathew, B., Léon, J., Sannemann, W., Sillanp, M.J., "Detection of epistasis for flowering time using bayesian multilocus estimation in a barley magic population," *Genetics 208 (2)*, 2018.

[21] Aflakparast, M., Salimi, H., Gerami, A., Dubé, M-P. Visweswaran, S., Masoudi-Nejad, A., "Cuckoo search epistasis: a new method for exploring significant genetic interactions," *Heredity (Edinb). 112(6)*, 2014.

[22] Tuo, S., Zhang, J., Yuan, X., Zhang, Y., Liu, Z., "Fhsa-sed: Two-locus model detection for genome-wide association study with harmony search algorithm," *PLoS ONE 11(3)*, 2016.

[23] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation 6(2): 182–197*, 2002.

[24] Han, B., Chen, X., Talebizadeh, Z., Xu, H., "Genetic studies of complex human diseases: characterizing snp-disease associations using bayesian networks," *BMC Syst. Biol. 6(Suppl. 3), S14*, 2012.

[25] Jiang, X., Neapolitan, R.E., Barmada, M.M., Visweswaran, S., "Learning genetic epistasis using bayesian network scoring criteria," *BMC Bioinform. 12(1), 89*, 2011.

[26] Wu, T.T., Chen, Y.F., Hastie, T., Sobel, E., Lange,K., "Genome-wide association analysis by lasso penalized logistic regression," *Bioinformatics 25(6)*, 2009.

[27] Akaike, H., "Information theory and an extension of the maximum likelihood principle," *Parzen E., Tanabe K., Kitagawa G. (eds) Selected Papers of Hirotugu Akaike*, 1998.

[28] North, B.V., Curtis, D., Sham, P.C., "Application of logistic regression to case-control association studies involving two causative loci," *Human Heredity 59(2): 79–87*, 2005.

[29] Yang, C., Wan, X., Yang, Q., Xue, H., Yu, W., "Identifying main effects and epistatic interactions from large-scale snp data via adaptive group lasso," *BMC Bioinformatics. 2010; 11(Suppl 1): S18*, 2010.

[30] Yang, C.H., Lin, Y.D., Chuang, L.Y., Chang, H.W., "Evaluation of breast cancer susceptibility using improved genetic algorithms to generate genotype snp barcodes," *Transactions on Computational Biology and Bioinformatics 10(2): 361–371*, 2013.

[31] Urbanowicz, R.J., Kiralis, J., Sinnott-Armstrong, N.A., Heberling, T., Fisher, JM, Moore, J.H., "Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures," *BioData Mining 5(1): 16*, 2012.