Published in final edited form as:

Comput Biol Med. 2008 July ; 38(7): 826–836. doi:10.1016/j.compbimed.2008.04.011.

A Parallel Genetic Algorithm to Discover Patterns in Genetic Markers that Indicate Predisposition to Multifactorial Disease

Tobias Rausch^{a,b}, Alun Thomas^a, Nicola J. Camp^a, Lisa A. Cannon-Albright^a, and Julio C. Facelli^{a,c*}

*a*Department of Biomedical Informatics, University of Utah School of Medicine, Salt Lake City, UT 84112, USA

*b*Hasso-Plattner-Institute, University of Potsdam, 14482 Potsdam, Germany UT 84112, USA

*c*Center for High Performance Computing, University of Utah, Salt Lake City, UT 84112, USA

Abstract

This paper describes a novel algorithm to analyze genetic linkage data using pattern recognition techniques and genetic algorithms (GA). The method allows a search for regions of the chromosome that may contain genetic variations that jointly predispose individuals for a particular disease. The method uses correlation analysis, filtering theory and genetic algorithms (GA) to achieve this goal. Because current genome scans use from hundreds to hundreds of thousands of markers, two versions of the method have been implemented. The first is an exhaustive analysis version that can be used to visualize, explore, and analyze small genetic data sets for two marker correlations; the second is a GA version, which uses a parallel implementation allowing searches of higher-order correlations in large data sets. Results on simulated data sets indicate that the method can be informative in the identification of major disease loci and gene-gene interactions in genome-wide linkage data and that further exploration of these techniques is justified. The results presented for both variants of the method show that it can help genetic epidemiologists to identify promising combinations of genetic factors that might predispose to complex disorders. In particular, the correlation analysis of IBD expression patterns might hint to possible gene-gene interactions and the filtering might be a fruitful approach to distinguish true correlation signals from noise.

Keywords

Gene-Gene Interactions; Multifactorial Diseases; Pattern Recognition; Data Mining; Correlation Analysis; Parallel Genetic Algorithm

1. Introduction

Many diseases and phenotypes are recognized to be multifactorial, that is, the result of several interacting genetic and environmental causes. Classical examples include asthma, hypertension or diabetes [1]. To elucidate the aetiology of complex, multifactorial diseases researchers must consider both genetic and environmental factors, as well as their interactions. Classical disease

*Corresponding author: Julio C. Facelli, Department of Biomedical Informatics, University of Utah School of Medicine, 26 South 2000 East - Room 5775 HSEB, Salt Lake City, UT 84112-5750, Phone: (801) 581-4080, FAX: (801) 581-4297, Email: Julio.Facelli@utah.edu.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

gene mapping methods [2], have been very successful in identifying the locations of genes predisposing to monogenic disorders, but for complex disorders these methods have been largely ineffective [1]. New strategies and methods that take into account interaction patterns are needed [3,4].

The extraction and discovery of useful information from large genetic data sets can be approached using pattern recognition and/or data mining methods [3–5]. Pattern recognition methods capitalize on computational intelligence techniques [6–10] or use metaheuristic optimization techniques like genetic algorithms (GA) to explore genetic data [11–15]. All of these methods emphasize the discovery of new information from the increasingly large statistical linkage data available to researchers. Moreover, many of these methods are well suited for parallel implementations that can exploit modern computer architectures and distributed computer systems [16–18], increasing their range of applicability.

The study of the genetics of complex disorders can be accomplished with association studies or with pedigree studies. In this paper we present a scalable pattern recognition approach to identify chromosomal regions of significance to complex diseases. While our methodology is general and can be easily extended to other experimental designs, we report in this paper its application to design studies that use pedigree based posterior IBD (identity by descent) probabilities for pairs of affected siblings. Our methodology can be used to identify common patterns in genotypes that can indicate presumption of causality to the disease by a definite genetic pattern. We present two implementations of our method, a comprehensive search for identifying two locus interactions in small data sets and a parallel genetic algorithm (GA) search that can be applied to multi-locus correlations in large data sets. We present results obtained with both implementations using simulated data of our own, and from the GAW 11 workshop [19].

2. Method

Our method is based on the following criterion to evaluate the plausibility that an observed pattern of specific markers indicates that a combination of genes in adjacent chromosomal regions is a determining factor in the onset of a disease:

If the IBD sharing for two, three, or n markers is very similar over all affected relative pairs, this is taken as an indicator that these two, three, or n markers are close in chromosome position to two, three, or n candidate genes that jointly affect the disease phenotype.

Given a pedigree structure, a marker map, and genetic marker data for pedigree members, we use Genehunter [20] to estimate posterior IBD values for each affected relative pair for all markers. The IBD values are entered into a matrix where each column corresponds to a marker and each row to one affected relative pair. The columns are ordered according to the physical position of markers along the chromosome and the markers should be equidistant or with distances interpolated by Genehunter or any other suitable program. Multiple chromosomes can be present in one matrix, as the end of a chromosome is easily recognized.

To assign a single value indicative of the sharing of genetic material by an affected relative pair (ARP) at the location of a given marker we use the posterior IBD distribution in the following sense:

$$\text{Cell Value} = 1 - P(\text{IBD}=0|\text{ARP}) = z \in [0,1] \quad (1)$$

The closer z is to 1, the more genetic material both individuals likely share at the given marker. In order to avoid assuming a recessive or dominant disease model at a specific locus we only use the posterior probability for "no sharing," but any other inheritance model can be used if

desired. An example of this matrix is shown in Table 1. Our stated criterion for marker association translates into seeking correlations among column vectors, where each column vector represents the IBD expression pattern for a certain marker. This approach extends naturally to higher order gene-gene interactions and in principle it could be used for unbounded searches. The literature in cluster analysis clearly shows that this is not feasible [21] in most cases, therefore we limit searches to correlations between N column vectors at a time.

The search space for correlation of N columns in a matrix with x columns consists of $\binom{x}{N}$ distinct sets of size N . In Table 2 the size of the search space is given for different N and matrix dimensions. The table shows that it is not feasible to explore the search space exhaustively, at least for $N > 2$. Consequently, we have implemented an exhaustive search method for $N = 2$, and a parallel genetic algorithm (GA) search method for $N \geq 2$.

There are numerous well-known correlation coefficients including Pearson's product moment correlation coefficient, Spearman's rank correlation coefficient, and various similarity coefficients used mainly in information retrieval [22]. For our method the vector correlation, or cosine coefficient, proposed as a model for information retrieval in 1975 [23], appears the most adequate because it emphasizes the high value components of a vector and tends to disregard small values that may be due to poor signal to noise common in many genotyping experiments. The cosine coefficient among two vectors is defined as:

$$\cos(\angle(\vec{x}, \vec{y})) = \cos(\angle(\vec{y}, \vec{x})) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|} = \frac{\sum x_i \cdot y_i}{\sqrt{\sum x_i^2} \cdot \sqrt{\sum y_i^2}} \quad (2)$$

It is used in this work to assess the similarity, i.e. the correlation, between two columns in the matrix.

Due to their physical proximity, or linkage [2], markers situated physically close to each other on a chromosome are transmitted together from parents to offspring more often than expected by chance. Consequently, the IBD sharing pattern will be very similar among physically close markers. In the search for gene-gene interactions relevant to complex diseases these naturally occurring marker correlations are of no relevance. Consequently, some kind of distance correction has to be applied to the chosen similarity statistic given by the vector correlation in eq. (2). Due to a lack of appropriate models to simulate recombination realistically, different distance correction functions could be considered. The only requirement for all of them is that the correlation-correction increases with the physical proximity of the two marker loci. The following distance correction function, that fulfils this requirement, was chosen for this study:

$$\text{Distance Correction}(\vec{x}, \vec{y}) = \begin{cases} 1 - \frac{\text{dist}(\vec{x}, \vec{y})}{\text{MaxDist}} & , \text{dist}(\vec{x}, \vec{y}) \leq \text{MaxDist} \\ 0 & , \text{otherwise} \end{cases} \quad (3)$$

The function $\text{dist}(x, y)$ returns the actual physical distance between two markers and MaxDist is a parameter that specifies the maximum distance over which the distance correction is applied. According to Haldane's mapping function [24] two loci that are 50 cM apart exhibit a recombination fraction of 32%. Tight linkage is observed for loci closer than 10 cM, and if two loci are inherited independently the recombination fraction equals 50% and the cM distance between them is infinite by definition. Hence, the parameter MaxDist should be ≥ 50 cM because at this distance natural correlations should be very small. A value of 75 cM was used in this study. The effect of this distance correction can be seen in Fig. 1(b).

Current genome-scans may use hundreds or hundreds of thousands of markers [25,26]. As a result, it is expected that some markers will show correlation by chance. Another important

consideration in our method is that genotype data may contain significant amounts of noise due to the experimental techniques used and the lack of specificity of gene-gene interactions. Consequently, suitable mechanisms to distinguish random correlations from true correlations must be used. In linkage analysis, markers may not be located within the disease gene. However, it is expected that markers which are physically close to a disease locus will co-segregate with the hypothetical disease locus. This co-segregation is by no means limited exclusively to the closest neighboring marker. On the contrary, it is expected that this co-segregation will extend to close flanking markers up and down the chromosome. Hence, the observed correlations between markers close to disease genes should extend over a chromosomal region, while random correlations are not expected to extend to neighboring markers to the same extent. Thus, averaging over the neighborhood of a candidate solution in the search space should be a suitable approach to separate noise from a true signal, i.e. extracting meaningful correlations. Researchers in the field of digital signal processing have developed various filtering methods to reduce noise in all kinds of signals. Well-recognized smoothing filters include linear filters, like the mean filter and the gauss filter, and non-linear filters like the median filter. For our data smoothing or averaging filters have proven the most useful in a limited number of test cases and were used throughout the study.

3. Exhaustive Search Method

The exhaustive search method uses four processing steps to discover candidate susceptibility loci:

1. Calculate the cosine coefficient for all possible combinations of two column vectors (markers)
2. Apply a distance correction to attenuate correlations among neighboring markers
3. Average over the neighborhood of a two marker association, i.e. filtering
4. Identify clusters of markers with the highest correlation

The exhaustive search algorithm pre-computes the cosine correlation coefficient among all possible pairs of markers and thus, it can be visualized as depicted in Fig. 1. In this figure the x-coordinate corresponds to the first marker and the y-coordinate corresponds to the second marker. The gray value of each pixel is set according to the correlation value, i.e. the higher the correlation the brighter the pixel. The image is mirrored along the diagonal because of the symmetric nature of correlations.

In the case of two marker correlations, filtering can be achieved by classical image filtering. In the current version of the software a mean and a median filter of user-defined size have been implemented. The effect of applying these filters to the search space of two marker correlations is shown in Fig. 2. In both images we applied a threshold to get a binary image that highlights the noise reduction. The threshold was chosen so that only the highest 1% of all correlation values is shown. Note how in the filtered image, Fig. 2(b), multiple high correlation values are clustered together.

Cluster detection is easily achieved by first setting a threshold to binarize the image and subsequently, running a labeling algorithm over the binary image. The labeling algorithm attaches a unique identifier to each distinct cluster. Based on these labels the x-range and y-range of every cluster can be determined. In a final processing step the highest peak in every cluster is identified. According to the size of this peak, all clusters are ordered and written to a file.

3.1. Results

The exhaustive search method has been applied to data sets provided for the Genetic Analysis Workshop 11 (GAW 11) [19] and to synthetic data sets created with the extended version of SIMLA [27], which is able to simulate gene-gene interactions. With SIMLA we simulated genetic data for families using the following parameters:

1. *Number of families*: a total of 800 nuclear families with a sibship size of 2 were simulated.
2. *Number of chromosomes and markers*: one chromosome (autosome) was simulated with 2 interacting disease loci and 250 random micro satellite markers. All markers were 2 cM apart and each one had 7 alleles with random allele frequencies.
3. *Disease loci*: each disease locus had two alleles, one disease allele and one normal allele. No linked haplotypes, environmental covariates or markers in linkage disequilibrium were simulated. The disease loci, called A and C, have been placed at 1.66 Mb and 3.33 Mb, respectively.
4. *Disease penetrance, disease prevalence and interaction*: SIMLA requires a parameter vector to specify disease penetrance, disease prevalence and strength of gene-gene interaction. The chosen values of this parameter vector are given below. The disease allele is denoted as D and the disease locus as 1 or 2 respectively:
 - $P(D1)=0.2, P(D2)=0.2$
The susceptibility allele frequency for both disease loci was set to 0.2
 - $W(D1)=0.77, W(D2)=0.77$
The weight for the genotype "Dd" was set to 0.77 because in complex disorders clear dominant or recessive disease models are usually not observed
 - $RR(D1,D1)=1.87, RR(D2,D2)=1.87$
The marginal effect of both loci was set to 1.87
 - $RR(G1 \times G2)=2.58$
To emphasize the gene-gene interaction the relative risk for the interaction was set to 2.58
 - $P(\text{Affected})=0.05$
The disease prevalence was set to 0.05

For each replicate SIMLA creates a pedigree file, a marker file and a map file. Using MEGA2 [28] these files were transformed into a format amenable to Genhunter, which was used to estimate and interpolate the IBD values every 1 cM for each affected relative pair. A post processing tool reads the IBD output file produced by Genhunter to generate the required matrix using eq. (1).

A total of 10 data sets were generated using the procedure outlined above. All 10 matrices had 800 rows (800 affected sib pairs) and 500 columns (Chromosome's length of 500 cM). Consequently, disease locus A and C are expected to occur around column 166 and 333 respectively.

The exhaustive search algorithm was run on all 10 data sets. A filter queue of a median filter followed by a mean filter was used to reduce the noise in the search space. The filter mask size was set to 5. The clusters found in the first data set are shown in Table 3. The second cluster

corresponds to the true correlation. The center of that cluster is, however, a few cM distant from the expected peak location. This might be an artifact of the Genehunter IBD estimation, because the problem of shifted peaks has been noted by other authors [29]. The results from the exhaustive search over all 10 data sets are summarized in Table 4, where the entries denote the cluster rank where the disease locus or the true correlation first appeared in a given data set. The true correlation was discovered in 9 out of 10 data sets and both disease loci were found in all data sets. One drawback of the proposed method is that true disease loci were frequently found to be associated with other markers as well (false positives). Interestingly these false positive correlations tended to occur more often for disease loci with a highly significant ($p < 0.01$) non-parametric linkage (NPL) score [20].

In the GAW 11 data set 4 disease loci called A, B, C, and D were present. A and B act in an interactive manner and locus D has a susceptibility allele that increases the risk of being affected by the two-locus type (A and B). Locus C can cause the disease independently of the other disease loci. In addition to discovering the disease loci, the correlation of A+B must be identified. According to the summary of previous analyses for GAW 11 [30] locus C was easy to find. The discovery of the disease loci A and B and the discovery of the gene-gene interaction was, however, a major problem for all research groups, mainly because of the small sample size in each replicate. In Table 5 we report the analysis results of the 25 replicates provided in the workshop's data set. Analogous to the SIMLA results, all the numbers in this table denote the cluster rank where the disease locus or the true correlation first occurred. We also included the A+D and B+D interaction to show how the method responds to susceptibility alleles.

Our results are in-line with the GAW 11 workshop summary presented in [30]. Investigators participating in the workshop easily detected locus C and this locus also appears frequently in one of the best clusters in our analysis. Locus D was easy to detect with an association test (e.g., transmission disequilibrium test) but due to the very high relative risk of locus D it could also be detected in a linkage-based analysis. This might be the reason why locus D is also present in a number of clusters in our analysis. Our results also confirm the fact that locus A was harder to detect than locus B because locus A appears in less clusters than locus B.

4. Parallel Genetic Algorithm Method

The large size of the search space for gene-gene interactions among more than 2 loci or for data sets with a large number of markers requires the use of non-exhaustive search strategies. Genetic algorithms (GA) [31] are a natural choice because we have extensive experience [32] in their parallel implementation.

Genetic algorithms are population-based search techniques. Every population member, hereafter called an individual, is a candidate solution to the problem. The basic structure of a genetic algorithm is shown in the flowchart of Fig. 3. For the proposed method a very intuitive encoding for an individual is a binary word, i.e. a sequence of zeros and ones. Every bit in this sequence corresponds to a column in the matrix given by eq. (1). If a bit is set to 1 the corresponding column is selected for correlation analysis. If a bit is set to 0 the column is not selected. The length of an individual (i.e., the length of its binary word) is equal to the number of columns in the matrix. The number of 1s in an individual's encoding is dependent on the order of correlations a researcher is looking for. For example when correlations among 4 markers are searched in a matrix with 10 columns, two possible individuals, called A and B, might look like this:

A: 0010110100
B: 0100101100

The fitness calculation of an individual subsumes the computational steps described for the exhaustive variant. In principle, the more fit an individual is, the better its encoded columns are correlated. Since the cosine coefficient estimates the angle between vectors, a logical way to calculate the correlation between 3, 4, or n markers is to average the pair-wise cosine correlation coefficients. The distance correction for every pair-wise correlation is applied in exactly the same manner as described for the exhaustive search. The ascertainment of neighbors for filtering is the most difficult step in the GA search. For illustrative reasons, it is helpful to start with only one dimension (one selected bit). In this case the neighbors can be ascertained by shifting the selected bit to the left and right. If we arrange all individuals on a line, the neighborhood forms a line segment which has the length of the filter size. In the two dimensional case all neighbors form a square if individuals are arranged on a grid. Note that again all neighbors can be ascertained by shifting one or both bits to the left and right. Obviously, this is the analogous case for the previously described image filtering for the exhaustive search implementation. In three dimensions the neighbors form a cube and they can still be ascertained by shifting individual bits of the encoding. This procedure can be extended to arbitrary dimensions. Consequently, the search space can still be filtered even for orders of correlations greater than three using the same approach used for the exhaustive search implementation. Once the neighbors of an individual have been ascertained a local averaging can be carried out. Note that we do not filter the complete search space because the GA search just samples candidate solutions in the search space.

The fitness calculation is, however, time-consuming due to correlation analysis, ascertainment of neighbors and filtering. A filter queue of multiple filters actually requires a several fold ascertainment of neighbors for correct filtering. To improve efficiency of the GA search a parallel version has been implemented that distributes the fitness calculations to several individual processors. The speed-up of this parallel version and the implementation details are described below.

A simple tournament selection schema is used to select individuals as parents for the next generation. Parents are crossed during reproduction using a modified one point crossover. The modification was necessary to guarantee that all new individuals still have the same amount of 1s in their encoding, i.e. they correlate the same number of markers. To meet this constraint we first determine all possible crossing sites as proposed in [33] and subsequently, we choose one of the candidate sites at random. A simple example is given below where all possible crossing sites have been marked with a vertical line.

```

A: 0|01|0|1|10|1|0|0
B: 0|10|0|1|01|1|0|0
After a crossover using the second site:
A*: 010|0110100
B*: 001|0101100

```

The second reproduction operator implemented is a mutation operator. During mutation an individual is selected at random out of the whole population, two non equal bits in the individual are also randomly selected and the operator simply flips one true bit and another false bit. The rate of both reproduction operators and the population size can be adjusted.

At the end of the reproduction phase the new generation is completed and the genetic algorithm performs the next iteration. If the fittest individual remains the same for more than 50 iterations the algorithm creates random individuals. These random immigrants replace less fit individuals in the population to avoid premature convergence to local extrema. The genetic algorithm is stopped after a predefined number of iterations.

4.1 Implementation of the parallel version

The calculation of the correlation coefficient, the ascertainment of neighbors and especially the filtering over the neighborhood turned out to be computationally demanding. Since these operations are carried out in the fitness evaluation function, it is expected that the fitness calculation contributes significantly to the overall running time of the GA. This was confirmed using the GNU profiler, gprof, [34], which is an easy to use and widely available debugging tool that analyses how much time is spent in which subroutine of a complex program and how often the subroutines are called. In a GA run with a population of 90 individuals the fitness evaluation function was called 900 times in 10 iterations, consuming the largest fraction of computer time. There is consistent agreement in the literature that the farmer-worker or master-slave model is the preferred method to parallelize GA[35]. Two versions of the farmer-worker model that allow an efficient parallelization of the fitness calculation have been proposed, the synchronous and the asynchronous farmer-worker models [36]. We have adopted the former, which is simpler to implement, because in our case it is reasonable to assume that the fitness evaluation time is approximately the same for all individuals. All the individuals encode the same amount of selected columns, ascertain the same number of neighbors for filtering and perform the same filtering functions. For each iteration of the GA, the number of independent tasks, i.e. calculation of the fitness functions, is equal to the population size and the communications are limited to the distribution of the individuals and the collection of their fitness values. In our implementation the farmer distributes the individuals and collects the fitness results. Workers receive individuals, evaluate the fitness, and send the fitness results back to the farmer. The mapping of tasks to processors is basically a one to one mapping; however, if the number of processors is less than the number of individuals, each processor can receive multiple individuals.

We have implemented the parallel GA code using the Message Passing Interface (MPI) library specification [37]. The current version of our parallel program uses MPICH 1.2.7, which offers full support of the MPI-1.1 standard. Because the pattern of communication is the same for each worker, it lends itself to collective communication operations, namely MPI_Bcast, MPI_Scatter, and MPI_Gather, instead of MPI_Send and MPI_Recv. During a comparative test described later in this paper, the implementation based upon collective communication operations proved to be superior to regular MPI_Send and MPI_Recv calls.

4.2. Results

None of the genetic simulation software packages known to the authors is capable of simulating interactions among three or more loci and we are not aware of any synthetic data sources with such high-order interactions. To circumvent the lack of appropriate simulation tools we decided to mimic a three loci interaction by simulating all the pair-wise interaction patterns among 3 loci. Hence, we decided to extend the simulation of locus A (at 1.66 M) and locus C (at 3.33 M) from the exhaustive search to include another disease locus B (at 2.50 M). For all pair-wise interactions (A+B, A+C, and B+C) we simulated a matrix in exactly the same manner as described before for the exhaustive search. Hence, each pair-wise data set yielded a matrix of 800 rows and 500 columns. We combined these three matrices into a single matrix with 2,400 rows and 500 columns by simply attaching the pair-wise matrices top to bottom.

In total we created 10 matrices where each matrix consisted of three pair-wise sub matrices. The parallel genetic algorithm was configured to search for three loci interactions and to use a median filter of size 5. The results are summarized in Table 6. The expected three loci correlation was discovered in half of the data sets. Interestingly, a single filter of small size gave the best results whereas large filters tended to erase true correlations. This argues in favor of the hypothesis that the search space corresponding to higher order correlations is less noisy. In other words, random correlations among three loci are less likely than among two loci. For

real data this might indicate that a search for gene-gene interactions among more than two loci may be actually easier than a search for two locus interactions. The search space is, however, substantially larger, and non-enumerative parallel algorithms are required to secure the computer capacity needed to perform these massive searches. While the success ratio for finding the three way correlation was low (50%), the results in Table 6 indicate that the method fulfills its data mining role of identifying promising correlations that should be further explored.

4.3 Performance Characteristics of the Genetic Algorithm

For GA two interesting questions arise naturally in terms of performance, namely convergence behavior and effectiveness in the exploration of the search space [38]. Convergence relates to two issues, whether the algorithm produces the same result in different runs and how long the algorithm needs to converge. The convergence behavior can be easily tracked in two dimensions where we can compare the results to our exhaustive implementation. As a test case we used the first replicate file of the GAW 11 data set [30]. The highest peak found by the exhaustive algorithm is of 0.885875 at position (714, 1170). The GA was executed 10 times on the same data set, with the following settings: population size = 140, maximum number of iterations = 1000, crossover-rate=0.7, and mutation-rate=0.4. Because GA are a stochastic search method their convergence criteria are not well defined. Commonly GA are run either for a fixed number of generations, a finite computer time or until the population reaches some predefined level of stagnation [31,32]. In this work we have chosen a fixed number of generations, i.e. 1000. All other parameters like the filter queue, the order of correlations, or the distance between chromosomes were the same as those used in the exhaustive search. In Table 7 the results of these 10 runs of the GA are summarized. In 80% of the runs the highest peak was found and in the two other runs a value close but different to the peak was identified. This value is still within the highest cluster and just 1cM distant from the highest peak.

The number of iterations fluctuates significantly. This makes it difficult to implement an upper bound on the number of iterations with the same highest fit individual to stop the GA automatically. Note that by using random immigrants the algorithm sometimes improves a candidate solution even after hundreds of iterations of stagnation. In general the time to convergence may also depend on the population size, but this dependence has not been studied here. Provided that enough resources are available, it is always advisable to run any metaheuristic search algorithm multiple times to test whether results are stable or not. If the results are not stable, increasing the upper bound for the number of iterations or increasing the population size might solve the problem.

To analyze the effectiveness of the GA with respect to the number of evaluation of the fitness function required, we must compare with those needed for an exhaustive search. For our example with 2,424 markers and with disregard of the order of columns there are nearly three million ways (2,936,676) of picking two distinct columns out of 2,424 candidate columns. A random search thus needs on average 1,468,338 evaluations of candidate solutions to find the optimum. In the above 10 runs of the GA the average number of iterations to discover the best solution was only 390. With a population size of 140 individuals the GA evaluated at most 54,600 individuals in 390 iterations, this represents a major improvement in comparison to a random search.

To assess how well our GA explores the solution space an example data set containing two clusters was used. The binary image showing these two clusters is reproduced in Fig. 4. As the GA progresses, more and more individuals should be close to these two clusters. That is, points in the search space close to the given clusters should be sampled at a higher frequency than points far away from these clusters. This is clearly demonstrated in Fig 4. After the initialization the sampled points are distributed randomly throughout the search space, Fig 4(a), but after 400 iterations more and more of the sampled points are close to the final clusters. In Fig. 4 (d)

the clusters can already be detected by simple visual inspection of the regions with the most sampled points. Note that the lines appearing in the image, which can be eliminated using our filtering techniques as demonstrated in Fig. 2, can be attributed to an artifact of the mutational operator. A good candidate solution in two dimensions is mutated by keeping one coordinate constant and moving the other one to the left and right. This naturally leads to a line pattern in a region of good candidate solutions.

4.4. Efficiency of the Parallel Implementation

Two profiling tools were used to analyze our parallel GA implementation, TAU (Tuning and Analysis Utilities) [39] and VAMPIR (later called Intel Trace Analyzer) [40]. TAU can automatically instrument parallel source code and it generates trace files of the program execution. VAMPIR is a trace file analyzer that enables users to analyze the time the program spent in calculation, input/output, and MPI routines. As mentioned above, the farmer-worker model has been implemented twice, once using MPI_Send and MPI_Recv operations, and once using MPI collective communication operations. The speed-up [36] of these two implementations was analyzed in a series of runs with varying population sizes and different numbers of GA iterations using one replicate file from the GAW 11 data set [30]. Every configuration of the GA was executed sequentially on one processor and in parallel on 4, 8 and 16 processors for both implementations of the farmer worker model. The speed-ups achieved are given in Table 8 and Table 9.

A simple comparison between the average speed-up achieved for the MPI_Send and MPI_Recv implementation with the average speed-up achieved in the MPI collective operations implementation shows clearly that the latter is superior in terms of efficiency. This is true independent of the actual number of processors used for the parallelization. Both implementations scale reasonably well. When using the second configuration and the collective operations implementation, a super linear speed-up of 4.2 and 8.1 was achieved on a 4 and 8 processor run. Super linear speed-ups can occur if the hardware favors a parallel implementation, but in this case we argue in favor of an outlier due to multiple jobs accessing the same scratch space during the runs. Nevertheless, even if row 2 is considered as an outlier the average speedup of the collective operations implementation is still superior to the implementation using MPI_Send and MPI_Recv calls. In general, all results argue in favor of the simple synchronous farmer-worker model. For example, a 4 processor version has an ideal speed-up of 4, while the current implementation achieved an average speed-up of 3.9 which does not leave much leeway for further improvement.

5. Discussion and Future Work

The method presented in this paper was motivated by the fact that classical linkage methods become complex and intractable when considering multi gene disease models. The proposed method is, however, not an attempt to substitute established linkage statistics but rather to use linkage information from a pattern recognition perspective. This is accomplished by separating the calculation of the linkage information and then using pattern recognition to search for possible multidimensional correlations in the data sets containing the linkage information. The method is neither restricted to a specific linkage statistic nor to a certain optimization technique in the metaheuristics variant. For instance in case-control studies one could use the identity by state allele sharing measure instead of the IBD statistics for the calculation of the probabilities in eq. (1) or implement a non enumerative search algorithm different than the GA.

The results presented for both variants of the method show that it can help genetic epidemiologists to identify promising combinations of genetic factors that might predispose to complex disorders. In particular, the correlation analysis of IBD expression patterns might

hint to possible gene-gene interactions and the filtering might be a fruitful approach to distinguish true correlation signals from noise.

It should be emphasized again that the proposed method is quite general and it is not limited to a particular inheritance model or statistic. As far as the experimental design allows calculation of the probabilities in eq. (1) for a set of markers, the method can be used to identify the multi dimensional correlations present in the matrix. The method also could be enhanced by using a more realistic method to estimate the distance penalty given by eq. (3), when such models emerge from detailed experimental studies of recombination probabilities [41]. Finally the inclusion of environmental factors in the analysis is immediate; as such factors can be included as additional columns in the matrix.

Nevertheless, the experiments on synthetic data sets revealed some limitations. First, the method requires a dense marker map because otherwise the filtering only affects interpolated marker data, this appears to be a modest limitation as high density genotyping becomes more affordable. Second, although the current method design can incorporate environmental factors by means of adding columns for their representation it is not clear how to treat these columns during filtering and how to assess the similarity of affected relative pairs for an environmental factor on a scale that is comparable to the linkage statistic, this issues will require further investigation using both simulated and experimental data. The convergence of the GA method and the option of using alternative global optimization methods should be also considered worthy of further investigation.

Acknowledgements

Calculations presented in this paper were performed on the Center for High Performance Computing Arches metacluster that has been partially funded by the National Institutes of Health (Grant #NCRR 1S10RR17214-01). The simulated data sets provided for the GAW 11 were supported by the grant DK31775 and GM31575, also from the National Institutes of Health. Alun Thomas was supported in part by NIH NIGMS grant R21 GM070710.

References

1. Mayeux R. Mapping the new frontier: complex genetic disorders. *The Journal of Clinical Investigation* 2005;115:1404–1407. [PubMed: 15931374]
2. Ott, J. *Analysis of Human Genetic Linkage*. Baltimore: The Johns Hopkins University Press; 1999.
3. Costello TJ, Falk CT, Ye KQ. Data Mining and Computationally Intensive Methods: Summary of Group 7 Contributions to Genetic Analysis Workshop 13. *Gen. Epidem* 2003;25:S57–S63.
4. Cupples LA, Bailey J, Cartier KC, Falk CT, Lui K-Y, Ye Y, Yu R, Zhang H, Zhao H. Data Mining. *Gen. Epidem* 2005;29:S103–S109.
5. Bishop, CM. *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press; 1995.
6. Firneisz G, Zehavi I, Vermes C, Hanyecz A, Frieman JA, Glant TT. Identification and quantification of disease-related gene clusters. *Bioinformatics* 2003;19:1781–1786. [PubMed: 14512349]
7. Gamberger D, Lavrač N, Železný F, Tolar J. Introduction of comprehensive models for gene expression datasets by subgroup discovery methodology. *J. Biomed. Informatics* 2004;37:269–284.
8. Lucek P, Hanke J, Reich J, Soll SA, Ott J. Multi-locus nonparametric linkage analysis of complex trait loci with neural networks. *Hum. Hered* 1998;48:275–284. [PubMed: 9748698]
9. Roth V, Lange T. Bayesian Class Discovery in Micorarray Datasets. *IEEE Trans. On Biomed. Eng* 2004;51:707–718.
10. Sherriff A, Ott J. Applications of neural networks for gene finding. *Adv. Genet* 2001;42:287–297. [PubMed: 11037328]
11. Jourdan L, Dhaenens C, Talbi E-G. A data mining approach to discover genetic and environmental factors involved in multifactorial diseases. *Knowledge-Based Systems* 2002;15:235–242.
12. Jourdan, L.; Dhaenens, C.; Talbi, E-G. *Encyclopedia of Data Warehousing and Mining*. Idea Group; USA: 2005.

13. Shah S, Kusiak A. Cancer gene search with data-mining and genetic algorithms. *Computers in Biol. and Medicine* 2007;37:251–261.
14. Vermeulen-Jourdan L, Dhaenens C, Talbi E-G. A Parallel adaptive GA for linkage disequilibrium in genomics. 2004
15. Vermeulen-Jourdan L, Dhaenens C, Talbi E-G. Linkage disequilibrium study with a parallel adaptive GA. *IJFCS: International Journal of Foundations of Computer Science* 2005;16:241–249.
16. Berman F, Dunning T. Designing and Supporting Science-Driven Infrastructure. 2006
17. Li WW, Baker N, Baldrige K, McCammon JA, Ellisman MH, Holst M, McCulloch AD, Michailova A, Papadopoulos P, Olson A, Sanner M, Arzberger PW. National Biomedical Computation Resource (NBCR): Developing End-to-End Cyberinfrastructure for Multiscale Modeling in Biomedical Research. *CTWatch* 2006;2:6–17.
18. Meyer F. Genome Sequencing vs. Moore's Law: Cyber Challenges for the Next Decade. *CTWatch* 2006;2:20–23.
19. Greenberg DA, MacCluer JW, Spence MA, Falk CT, Hodge SE. Simulated data for a complex genetic trait (problem 2 for GAW11): how the model was developed, and why. *Genet. Epidemiol* 1999;17:449–459.
20. Kruglyak L, Daly MJ, Reeve-Daly MP, Lander ES. Parametric and nonparametric linkage analysis: a unified multipoint approach. *Am. J. Hum. Genet* 1996;58:1347–1363. [PubMed: 8651312]
21. Salvador S, Chan P. Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms. 2004
22. Baeza-Yates, R.; Ribeiro-Neto, B. Modern Information Retrieval. New York: Addison-Wesley; 1999.
23. Salton G, Wong A, Yang ACS. A vector space model for automatic indexing. *Communications of the ACM* 1975;18:229–237.
24. Haldane J. The combination of linkage values, and the calculation of distances between the loci of linked factors. *J. Genet* 1919;8:299–309.
25. Gretarsdottir S, Sveinbjornsdottir S, Jonsson HH, Jakobsson F, Einarsson E, Agnarsson U, Shkolny D, Einarsson G, Gudjonsson HM, Valdimarsson EM, Einarsson OB, Thorgeirsson G, Hadzic R, Jonsdottir S, Reynisdottir ST, Bjarnadottir SM, Gudmundsdottir T, Gudlaugsdottir GJ, Gill R, Lindpaintner K, Sainz J, Hannesson HH, Sigurdsson GT, Frigge ML, Kong A, Gudnason V, Stefansson K, Gulcher JR. Localization of a susceptibility gene for common forms of stroke to 5q12. *Am. J. Hum. Genet* 2002;70:593–603. [PubMed: 11833004]
26. Kong A, Gudbjartsson DF, Sainz J, Jonsdottir GM, Gudjonsson SA, Richardsson B, Sigurdardottir S, Barnard J, Hallbeck B, Masson G, Shlien A, Palsson ST, Frigge ML, Thorgeirsson TE, Gulcher JR, Stefansson K. A high-resolution recombination map of the human genome. *Nat. Genet* 2002;31:241–247. [PubMed: 12053178]
27. Schmidt M, Hauser ER, Martin ER, Schmidt S. Extension of the SIMLA package for generating pedigrees with complex inheritance patterns: environmental covariates, gene-gene and gene-environment interaction. *Stat. Appl. Genet. Mol. Biol* 2005;4paper 15
28. Mukhopadhyay N, Almasy L, Schroeder M, Mulvihill WP, Weeks DE. Mega2: data-handling for facilitating genetic linkage and association analyses. *Bioinformatics* 21:2556–2557. [PubMed: 15746282]
29. Potash JB, Zandi PP, Willour VL, Lan T-H, Huo Y, Avramopoulos D, Shugart YY, MacKinnon DF, Simpson SG, McMahon FJ, DePaulo JRJ, McInnis MG. Suggestive linkage to chromosomal regions 13q31 and 22q12 in families with psychotic bipolar disorder. *Am. J. Psychiatry* 2003;160:680–686. [PubMed: 12668356]
30. Greenberg DA. Summary of analyses of problem 2 simulated data for GAW11. *Genet. Epidemiol* 1999;17:S429–S447. [PubMed: 10597474]
31. Goldberg, DE. Optimization and Machine Learning. New York: Addison-Wesley; 1989. Genetic Algorithms in Search.
32. Bazterra VE, Cuma M, Ferraro MB, Facelli JC. A General framework to understand parallel performance in heterogeneous systems. *J. of Parallel and Distrib. Comput* 2005;65:48–57.
33. Bazterra VE, Oña O, Caputo MC, Ferraro MB, Fuentealba P, Facelli JC. Modified genetic algorithms to model cluster structures in medium size silicon clusters. *Phys. Rev. A* 2004;69:053202.

34. Fenlason, J.; Stallman, R. GNU gprof. 2003 [Accessed April 6th 2008]. See also http://www.cs.utah.edu/dept/old/texinfo/as/gprof_toc.html
35. CantúPaz E. Genetic and Evolutionary Computation GECCO 2003 Part I. 2003
36. Grama, A.; Karypis, G.; Gupta, A. An Introduction to Parallel Computing: Design and Analysis of Algorithms. Addison Wesley; 2003.
37. Forum M. MPI: A message-passing interface standard. 1994
38. Haupt, RL.; Haupt, SE. Practical Genetic Algorithms. New York: Wiley-Interscience; 2004.
39. Windish K, Mohr B, Malony A. A brief technical overview of the TAU tools. 1996
40. Nagel WE, Arnold M, Weber M, Hoppe H-C, Solchenbach K. VAMPIR: Visualization and analysis of MPI resources. Supercomputer 1996;12:69–80.
41. Fearnhead P. SequenceLDhot: detecting recombination hotspots. Bioinformatics 2006;22:3061–3066. [PubMed: 17060358]Petes TD. Meiotic Recombination Hot Spots and Cold Spots. Nature Reviews Genetics 2001;2:360–369.

Biographies

Tobias Rausch was born in Berlin, Germany, in 1980. He received a Bachelor's degree and Master's degree in Software Engineering from the University of Potsdam, Germany, in 2005 and 2006, respectively. Since October 2006, he has been a PhD student at the International Max Planck Research School for Computational Biology and Scientific Computing. His research interests focus on sequence analysis and genome comparison.

Dr. Alun Thomas was born in Llandybie, Cymru in 1960. He received a degree from Coleg Prifysgol Cymru, Aberystwyth in 1981 and a PhD from Cambridge University in 1985. He was previously a lecturer in the School of Mathematical Sciences at the University of Bath, and Vice President of Bioinformatics at Myriad Genetics Inc. Since 2002 he has been a professor in the Department of Biomedical Informatics at the University of Utah. His research interests are in computational statistics applied in genetics.

Dr Camp was born in Southport, UK, in 1971. She received a first class honors degree in Mathematics from the University of Sheffield, UK, in 1992 and a PhD, from the same University, in 1995. Since 1998, she has been with the University of Utah, USA, where she is currently an Associate Professor in Biomedical Informatics (Division of Genetic Epidemiology). Her research interests focus on statistical genetics and gene identification in common cancers.

Dr. Cannon Albright received a BS in Statistics from Brigham Young University and a PhD in Medical Informatics from the University of Utah School of Medicine in 1988. She is currently Professor and Vice Chair, Department of Biomedical Informatics, University of Utah School of Medicine. She is a genetic epidemiologist with research interests in using genealogical resources to define the heritable contribution to health-related phenotypes and to identify predisposition genes responsible.

Dr. Facelli was born in Buenos Aires, Argentina and attended the University of Buenos Aires where he got his PhD in physics in 1982. He did post-doctoral research at the University of Arizona and the University of Utah. At Utah he is the Director of the Center for High Performance Computing, Professor of Biomedical Informatics and Adjunct Professor of Physics and Chemistry. His current research interest include parallel genetic algorithms for atomic cluster and crystal structure prediction, parallel applications to of genetic algorithms in bioinformatics and the practical use of Grid computing for solving complex problems.

(a) Image without Distance Correction

(b) Image with Distance Correction

Fig. 1.

Effect of applying the distance correction, eq. (3) to the correlation between all possible two marker correlations. The distance between markers is 1cM, MaxDist is 75cM. The six bars along the diagonal are due to the 6 chromosomes present in the data set. Distance correction does not cross chromosome boundaries and hence, these boundaries are visible in the image.

(a) Thresholded original image

(b) Thresholded filtered image

Fig. 2. Noise reduction and clustering achieved using the filters proposed in this study to eliminate random correlations. The highest 1% of all correlation values is shown in both images.

Fig. 3.
Flow diagram of the genetic algorithm (GA) used in this work.

Fig. 4.

Demonstration of the effectiveness of using a GA search for the exploration of a two dimensional correlation space with two clusters. Note that due to the symmetry of the correlations this image is mirrored along the diagonal. The lines observed in (c) and (d) correspond to artifacts due to the crossover operator in two dimension (see text).

Table 1
Example of the matrix, given by eq. (1), used to explore marker correlations. Ped is the pedigree identifier, Pair is the pair of identifiers of affected individuals, $z_{i,j}$ is the probability that pair i shares genetic material at marker j . Data from the GAW 11 [30], first replicate (Mycenaean study).

Ped	Pair	1	2	3	4	5	...
mv1	7,11	0.998608	0.997624	0.99704	0.99685	0.997052	...
mv2	4,5	0.99862	0.997593	0.996938	0.996673	0.99682	...
mv3	4,5	0.874967	0.751478	0.629149	0.507603	0.38647	...
mv4	3,5	0.998287	0.973336	0.978269	0.983055	0.987659	...
mv5	3,5	0.993712	0.989633	0.987574	0.98733	0.988688	...
mv6	4,6	0.946076	0.892973	0.840539	0.788625	0.737086	...
mv7	3,7	0.356834	0.353002	0.348702	0.343923	0.33865	...
mv8	4,5	0.999795	0.999693	0.999674	0.999716	0.999796	...
mv9	4,5	0.002794	0.004786	0.005979	0.006379	0.005986	...
mv10	3,4	0.99181	0.9849	0.97849	0.973683	0.968765	...
...	...	...	...	...	...	...	...

Table 2

Size of the search space for possible correlations between N possible columns in a matrix for selected total columns.

N	100 Columns	2,000 Columns	50,000 Columns
2	4 950	1 999 000	1 249 975 000
3	161 700	1 331 334 000	20 832 083 350 000
4	3 921 225	664 668 499 500	260 385 417 812 487 500

Cluster analysis of the results obtained using the exhaustive search method to explore the first data set generated using SIMLA (see text for the description of the data set generation). The disease loci present in each cluster are shown in the last column. Clusters are ordered according to peak size and characterized by their extension and center.

Table 3

Cluster Rank	x-Region	y-Region	Center	Peak Size	Disease Loci
1	70–98	164–187	89,175	0.796062	A
2	140–178	318–333	150,328	0.793378	A, C
3	71–93	316–331	89,325	0.79136	C
4	180–188	301–317	185,310	0.790633	-
5	87–91	294–298	89,297	0.789136	-

Table 4

Ranks of the first cluster including the disease locus or correlation of disease loci obtained using the exhaustive search method to explore all the data sets generated using SIMLA (see text for the description of the data set generation).

	A+C	A	C
Data set 01	2	1	2
Data set 02	4	1	4
Data set 03	1	1	1
Data set 04	6	3	4
Data set 05	2	2	1
Data set 06	4	1	3
Data set 07	-	2	5
Data set 08	5	1	5
Data set 09	1	1	1
Data set 10	1	1	1

Table 5 Summary of the results obtained analyzing the GAW 11 [30] data sets using the exhaustive search algorithm. The values in the table indicate first occurrence of disease locus or first occurrence of correlated disease loci in the data set.

	A+B	A+D	B+D	A	B	D	C
Mve01	7	12	8	2	7	4	1
Mve02			32		1	2	5
Mve03	7			2	1		1
Mve04		46	65	46	17	1	8
Mve05	2			1	2	8	3
Mve06		20	8	20	8	1	1
Mve07	17	29	2	17	2	2	3
Mve08	15			14	15	6	1
Mve09		49	36	24	3	2	1
Mve10				46	20		1
Mve11				14	40		1
Mve12			40		3	40	2
Mve13				48	4	8	1
Mve14	15	11	44	1	3	11	31
Mve15				28	7	9	1
Mve16				4	17		1
Mve17	44			44	4	42	18
Mve18			39		32	1	1
Mve19		14		14	6	10	2
Mve20	11			3	5	36	1
Mve21	2			2	2		1
Mve22					1	34	1
Mve23	20			3	10		1
Mve24	9	7		2	9	1	3
Mve25	60			2	60	35	61

Table 6

Summary of GA search results for simulated data sets including three way correlations. The data set was constructed as described in the text. The optimal correlation value (peak size) and the corresponding columns (optimal marker set) are shown for each data set. Disease loci identified by means of the GA search are shown in the last column.

Replicate File	Optimal Market Set	Peak Size	Disease Loci
Data Set 01	136 256 364	0.782358	B
Data Set 02	174 274 392	0.781859	A
Data Set 03	162 246 328	0.789289	A,B,C
Data Set 04	172 254 334	0.782352	A,B,C
Data Set 05	158 238 318	0.780947	A
Data Set 06	162 252 332	0.789418	A,B,C
Data Set 07	94 174 256	0.787214	A,B
Data Set 08	164 244 330	0.7832852	A,B,C
Data Set 09	84 168 396	0.777977	A
Data Set 10	168 248 336	0.785234	A,B,C

Table 7

Results for ten independent runs of the GA search for the first replica of the GAW 11 data set. The exhaustive search value is 0.885875 at position (714, 1170).

Run #	Peak Position	Peak Size	Number of Iterations
1	(714, 1170)	0.885875	789
2	(714, 1170)	0.885875	295
3	(714, 1170)	0.885875	262
4	(714, 1170)	0.885875	175
5	(714, 1171)	0.885809	387
6	(714, 1170)	0.885875	569
7	(714, 1170)	0.885875	360
8	(714, 1170)	0.885875	360
9	(714, 1170)	0.885875	550
10	(714, 1171)	0.885809	157

Table 8

Comparison of the different speedups achieved by the GA search using the implementation with MPI_Send and MPI_Recv.

Pop size, # Iterations	Number of Processors			
	1 ^a	4	8	16
80, 500	0:56:45	3.76	7.16	11.98
128, 500	1:29:48	3.898	7.29	13.10
160, 500	1:52:12	3.83	7.39	11.03
80, 1000	1:47:48	2.89	5.84	13.14
128, 1000	2:59:40	3.90	6.09	13.90
160, 1000	3:42:57	3.79	6.13	14.26
Average Speedup:		3.68	6.65	12.90

^a Elapsed time in HH:MM:SS.

Table 9

Comparison of the different speedups achieved by the GA search using the implementation with collective operations.

Pop size, # Iterations	Number of Processors			
	1 ^a	4	8	16
80, 500	0:57:06	3.87	7.32	13.17
128, 500	1:37:52	4.20	8.14	15.05
160, 500	1:51:31	3.81	7.38	11.67
80, 1000	1:53:07	3.87	7.49	13.93
128, 1000	2:58:39	3.82	7.51	14.14
160, 1000	3:43:45	3.85	7.54	14.59
Average Speedup		3.90	7.56	13.76

^aElapsed time in HH:MM:SS.