A FAST LASSO-BASED METHOD FOR INFERRING PAIRWISE INTERACTIONS

KIERAN ELMES, ASTRA HEYWOOD, ZHIYI HUANG, AND ALEX GAVRYUSHKIN✉

Abstract. Large-scale genotype-phenotype screens provide a wealth of data for identifying molecular alternations associated with a phenotype. Epistatic effects play an important role in such association studies. For example, siRNA perturbation screens can be used to identify pairwise gene-silencing effects. In bacteria, epistasis has practical consequences in determining antimicrobial resistance as the genetic background of a strain plays an important role in determining resistance. Existing computational tools which account for epistasis do not scale to human exome-wide screens and struggle with genetically diverse bacterial species such as *Pseudomonas aeruginosa*. Combining earlier work in interaction detection with recent advances in integer compression, we present a method for epistatic interaction detection on sparse (human) exome-scale data, and an R implementation in the package `Pint`. Our method takes advantage of sparsity in the input data and recent progress in integer compression to perform lasso-penalised linear regression on all pairwise combinations of the input, estimating up to 200 million potential effects, including epistatic interactions. Hence the human exome is within the reach of our method, assuming one parameter per gene and one parameter per epistatic effect for every pair of genes. We demonstrate `Pint` on both simulated and real data sets, including antibiotic resistance testing and siRNA perturbation screens.

1. Introduction

Epistatic gene interactions have practical implications for personalised medicine, and synthetic lethal interactions in particular can be used in cancer treatment [3]. Discovering these interactions is currently challenging, however. In particular, there are no methods able to automatically infer interactions from genotype-phenotype data at the human genome scale.

For a given number of genes there are exponentially many potential interactions, complicating computational methods. If we restrict our attention to pairwise effects, it is possible to experimentally knock out particular combinations of genes to determine their combined effect [10]. This approach does not scale to the approximately 200 million pairwise combinations possible among human protein coding genes, however. We instead consider inferring pairwise interactions from large-scale genotype-phenotype data. These include mass knockdown screens, in which we suppress a large number of genes simultaneously, and attempt to measure the resulting phenotypic effect.

We have shown in [12] that a lasso-based approach to inferring interactions from an siRNA perturbation matrix is a feasible method for large-scale interaction detection. In this additive model, we assume fitness is a linear combination of the effects of each gene's effect, and the effect of every combination of these genes. For the sake of scalability, we consider only individual and pairwise effects, and assume gene suppression is strictly binary. The fitness difference f (compared to no knockdowns) in an experiment e is then the sum of individual and pairwise effects $\sum_i^p g_i + \sum_i^p \sum_{j>i}^n g_i \cdot g_j$, where $g_i = 1$ if gene i is knocked down, 0 otherwise. With sufficiently many such mass-knockdowns, we can infer pairwise interactions by finding the pairs of genes whose effect is not the sum of the effects of each gene individually.

Department of Computer Science, University of Otago, New Zealand
E-mail addresses: ✉alex@biods.org.

We acknowledge support from the Royal Society Te Apārangi through a Rutherford Discovery Fellowship (RDF-UOO1702). This work was partially supported by Ministry of Business, Innovation, and Employment of New Zealand through an Endeavour Smart Ideas grant (UOOX1912) and a Data Science Programmes grant (UOAX1932).

Neither of the previously tested inference methods for this model, `glinternet` and `xyz`, are effective at the genome-scale however. `glinternet` suffers from prohibitively long running times,[1] and `xyz` does not accurately predict effects in our larger simulations. Our aim is to fit a model including all $p \approxeq 20,000$ human protein-coding genes, with as many as $n = 200,000$ siRNAs. Doing so requires the development of new methods and software.

We have developed an R-package that is able to perform lasso regression on all pairwise interactions on the same one thousand gene screen in twenty seconds, and is able to fit a genome-scale data set with $19,000$ genes and $67,000$ siRNAs in under two hours using a single eight-core CPU. This is made possible by taking into account that our input matrix X is both sparse and strictly binary. Our package, `Pint`, is available at github.com/biods/pint.

To perform lasso-based regression on this matrix, we begin with an existing fast algorithm, parallelise it, and adapt it for use on our binary perturbation matrices. We provide a detailed explanation of this implementation, followed by the scalability analysis, below. We also perform a simulation study to compare our method's scalability with known methods, and analyse two large-scale experimental data sets.

In the first, an siRNA perturbation screen from [29], we search for pairs of genes that have an epistatic effect when simultaneously silenced. Out of five top interactions identified by our method, two are known protein interactions and three appear to be novel.

The second data set is composed of genetic variants identified in the intrinsically antibiotic resistant bacteria *Pseudomonas aeruginosa*. *P. aeruginosa*is an opportunistic pathogen found in a variety of environments and is a leading cause of morbidity and mortality in immunocompromised individuals or those with cystic fibrosis [16, 22]. *P. aeruginosa*is known to acquire adaptive antibiotic resistance in response to long term usage of antibiotics associated with chronic infections [5, 25, 26]. The genomes included in that data set are from strains that have been isolated from chronic and acute infections as well as environmental samples. The minimum inhibitory concentration for the antibiotic Ciprofloxacin has been used as the phenotypic marker for this dataset. Ciprofloxacin belongs to the fluoroquinolone class of bacteriocidal antibiotics that targets DNA replication and is one of the most widely used antibiotics against *P. aeruginosa*[32]. Our findings identified 16 pairs of interactions, most of which were found in genes that are important in biofilm formation and maintenance, a characteristic of intrinsically antibiotic resistant bacteria.

2. Methods

Our goal is to estimate both the main effects $\beta_1, \ldots, \beta_p$, and the interaction effects $\beta_{1,2}, \ldots, \beta_{p-1,p}$ where pairs of genotypes are simultaneously perturbed. As an example, consider pairwise effects in a siRNA perturbation screen. We can estimate the effect of both silencing individual genes $(\beta_1, \ldots)$ and pairs of genes simultaneously $(\beta_{1,2}, \ldots)$. To do this we add a column for each pair of genes, converting the siRNA matrix $\mathbf{X} \in \{0,1\}^{n \times p}$ into the pairwise matrix $\mathbf{X}_2 \in \{0,1\}^{n \times p'}$, where $p' = \frac{p(p+1)}{2}$. This model includes all pairwise interactions and fitting it is equivalent to finding epistasis as in [12]. The same construction applies to any binary genotype-phenotype data, and the effect $\beta_{a,b}$ will always estimate the simultaneous effect of both genes a and b.

We construct the matrix $\mathbf{X}_2$ as follows. For every column i from 0 to n we take every further column j from $i+1$ to n and form a new column by taking the bit-wise *and* over all elements of the columns i and j (Fig. 1).

[1] Finding interactions in an siRNA screen of $1,000$ genes with ten siRNAs per gene takes several days using ten cores on an Opteron 6276.

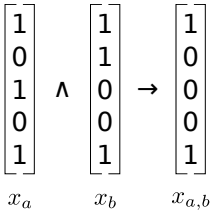


FIGURE 1. Creation of pairwise siRNA effect columns

This gives us the complete pairwise matrix $\mathbf{X}_2$, shown in Fig. 2.

Gene 1 Gene p Genes 1 and 2 Genes 1 and p Genes $p-1$ and p

$$\beta_1 x_{1,1} + \ldots + \beta_p x_{1,p} + \beta_{1,2} x_{1,1\wedge 2} + \ldots + \beta_{1,p} x_{1,1\wedge p} + \ldots + \beta_{p-1,p} x_{1,p-1\wedge p} + \mathscr{E}_1 = y_1$$

$$\beta_1 x_{2,1} + \ldots + \beta_p x_{1,p} + \beta_{1,2} x_{2,1\wedge 2} + \ldots + \beta_{1,p} x_{2,1\wedge p} + \ldots + \beta_{p-1,p} x_{2,p-1\wedge p} + \mathscr{E}_2 = y_2$$

$$\vdots$$

$$\beta_1 x_{n,1} + \ldots + \beta_p x_{n,p} + \beta_{1,2} x_{n,1\wedge 2} + \ldots + \beta_{1,p} x_{n,1\wedge p} + \ldots + \beta_{p-1,p} x_{n,p-1\wedge p} + \mathscr{E}_n = y_n$$

FIGURE 2. Matrix of Pairwise siRNA effects

2.1. **Cyclic Linear Regression.** Our approach to lasso regression is based on a cyclic coordinate descent algorithm from [15], as described in [38]. This method begins with $\beta_j = 0$ for all j and updates the beta values sequentially, with each update attempting to minimise the current total error. Here this total error is the difference between the effects we have estimated and the fitness we observe, given the genes that have been knocked down. Where y_i is the ith element of $\mathbf{Y}$, β_j is the jth element of β, and x_{ij} is the entry in the matrix $\mathbf{X}_2$ at column j of row i, the error is the following.

$$\sum_{i=1}^{n} |y_i - \sum_{j=1}^{p'} x_{ij} \cdot \beta_j| \tag{1}$$

The error affected by a single beta value (Eq. (5)) can then be minimised by updating β_k with the following:

$$\Delta\beta_k = \begin{cases} max(0, \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{S_k} - \lambda) & \text{for } \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{S_k} > 0 \\ min(0, \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{S_k} + \lambda) & \text{for } \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{S_k} < 0 \end{cases} \tag{2}$$

We cyclically update each β_k until the solution converges for a particular lambda, reduce the value of lambda, and repeat. See Appendix B for the full derivation and algorithm. Storing the matrix in a sparse column format, this implementation scales up to $p = 1,000$. It would still take several days and use terabytes of memory for $p = 20,000$. To overcome this, we compress the matrix, and parallelise the beta updates (Section 2.3 and Appendix A).

2.2. **Choosing Lambda.** The lasso penalty requires a regularisation parameter lambda. This parameter determines the extent to which we penalise large beta values, and can range from allowing all values ($\lambda = 0$) to allowing only zero ($\lambda \to \infty$). Choosing the correct value of lambda is essential if we want to include only the significant effects. This is typically done by choosing an initial value sufficiently large that all beta values will be zero and gradually reducing lambda, fitting the model for each value until a stopping point chosen with K-fold cross-validation [14]. Cross-validation requires fitting each lambda value K times, however, significantly increasing the runtime. We instead provide two options for choosing lambda in our package. First, we can

choose lambda such that the number of non-zero effects is small enough for OLS regression. In our package, this is called Limited-β and a default limit is $2,000$. Alternatively, we implement a fast method for empirically choosing a reasonable stopping point, the adaptive calibration lambda selection method from [8]. Both of these methods are significantly faster than cross-validation, although using adaptive calibration we tend to predict very few non-zero effects. The best empirical results are generally achieved with the Limited-β approach, and we use this for the remainder of the paper. A detailed explanation of each and their performance impact can be found in Appendix D.

2.3. **Compression.** To reduce memory usage and the time taken to read each column with larger input data, we compress the columns of $\mathbf{X}_2$. Because we read the columns sequentially, we replace each entry with the offset from the previous entry. This reduces the average entry to a relatively small number, rather than the mean of the entire row. These small integers can then be efficiently compressed with any of a range of integer compression techniques (Fig. 3), a subject that has been heavily developed for Information Retrieval. We compare a number of such methods, including the Simple-8b algorithm from [33] (which we implement and use in our package) in Appendix C.1.

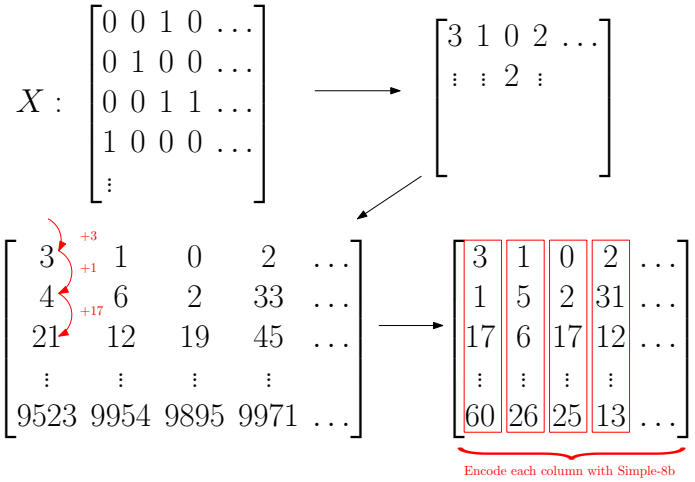
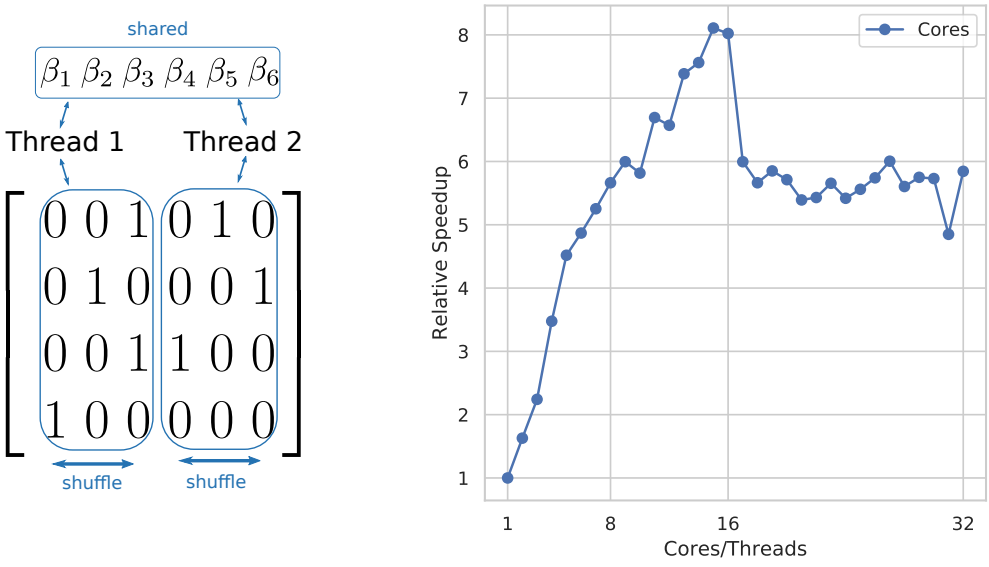


FIGURE 3. Compression of the sparse $\mathbf{X}_2$ matrix.

2.4. **Parallelisation.** While it is trivial to parallelise the update of a single β value, doing so does not improve performance in practice, due to poor cache usage (see Appendix A for details). We instead parallelise our method by assigning whole columns of the compressed $\mathbf{X}_2$ matrix to threads in sections. Each thread is responsible for updating the β values corresponding to its columns, and is therefore the only thread reading it's section of the $\mathbf{X}_2$ matrix.

Simultaneously updating columns with entries in the same row leads to over-compensating for these entries. This can harm performance or in the worst case prevent convergence entirely (Appendices A.1 to A.3). To avoid this, it suffices to ensure that threads do not frequently update the same columns at the same time. We achieve this by shuffling the order each thread updates it's columns every iteration. While it is in principle still possible to update enough overlapping threads in parallel to cause problems, Bradley et al. [6] show that this is rarely a problem in practice.

Updates to the shared β values are atomic, and every thread needs read access to all β values. This limits our method to use on shared memory systems and results in poor performance on NUMA systems. In practice we can only effectively use a single CPU socket, and we have tested this up to eight cores. Fig. 4 summarises the shuffled parallel implementation and it's scalability. For the full details of the parallel implementation see Appendix A.



(a) (b)

FIGURE 4. (a) Each thread is assigned a set of columns, which is then shuffled every iteration.
(b) Relative speedup as the number of cores used increases, running on a dual 8 core/16 thread NUMA system. Cores 1-8 are separate cores on node 1, 8-16 are SMT threads on the same cores. Cores 17-24 are separate cores on the second NUMA node, and 15-32 are SMT threads on those cores.

2.5. **Limited Interaction Neighbourhoods.** When searching for interactions within a large sequence, it may be acceptable to limit the search to pairs that are relatively close on the genome. In a study of epistatic interactions in yeast by Puchta et al. [28] the strength of negative interactions decreases as distance between gene positions on the sequence increases. The median distance between pairs in the hundred strongest interactions was only eighteen nucleotides.

Limiting interactions to those within some distance d drastically reduces both the time and space requirements. Instead of $\Theta(p^2n)$, the size of the interaction matrix becomes $\Theta(pdn)$. Similarly, an iteration of Algorithm 1 would require only $\Theta(pdn)$ operations. For $d << p$ this is a significant reduction. Limiting the interaction search distance to 100 positions, we could process a set of $30,000$ genes and $200,000$ siRNAs using approximately $16GB$ of memory, assuming a comparable density of interactions to our testing data. Such a search could be performed directly on a laptop, without requiring access to a large server. The biological implications of this restriction should be carefully considered before its use, however.

2.6. **Data.** We prepared two experimental data sets to evaluate our method and test the scalability of our implementation. The first is an siRNA perturbation screen in which siRNAs targeting kinases are applied to an infected human cell line. We predict off-target effects across the entire exome, and use this larger set for our analysis.

The second data set contains single nucleotide variants (SNVs) from 259 isolates of *Pseudomonas aeruginosa*, and associated minimum inhibitory concentration (MIC) of Ciprofloxacin.

2.6.1. *InfectX siRNA Data.* To demonstrate our method on real genome-scale data, we use the vaccinia group from InfectX [29]. This set contains $204,288$ siRNA perturbations in the presence of the vaccinia virus. This set is significantly larger than the mock group (siRNA perturbations with no pathogen present). Off-target effects are prediction using Risearch2 [1]. We include a gene as an off-target effect whenever there is a match between the siRNA seed region and some

component of an mRNA for that gene (taken from [17]). We use an energy cutoff of -20 and match the entire siRNA, not only the $3'$ UTR, as suggested in [1].

We then form a matrix of off-target effects with columns for each gene, and rows for each siRNA as in [12]. An entry i, j in this matrix is one if and only the predicted effect of siRNA i on gene j is greater than zero. All other entries are zero. Our fitness vector $\mathbf{Y}$ is the result of B-scoring then Z-scoring the number of cells in the well, to remove systematic within-plate effects and experimentally introduced cross-plate biases. B-scoring corrects for biases across the entire plate, and Z-scoring then normalises each well's score with respect to the rest of it's plate.

2.6.2. *Antibacterial Resistance.* SNVs from 259 isolates of *Pseudomonas aeruginosa*were sequenced using illumina technologies (IPCD isolates on MiSeq and QIMR isolates on HISeq). SNV's from raw reads were mapped to the reference genome PAO1 using Bowtie2 (v. 2.3.4) [20] read aligners. Variant reports were then read into a python script which sorted the reports into a table. The table was set up so that each isolate was represented as a row and the presence / absence of each SNV was along the columns. Only genomes that had associated MIC values were included. The resulting table contains 259 rows and over $700,000$ columns.

Since our method considers p^2 interactions, the scale of this data presents a problem. Including all $> 700,000$ columns, we would need to store over 250 billion interaction columns, each with up to 259 entries. Even if every column fits into a single 64-bit word, simply storing the compressed matrix would require on the order of two terabytes of memory. We instead reduce this to a more manageable scale, by removing all duplicated columns, and then any of the remaining columns that have less than 30 entries. Note that this is likely to remove point mutations occurring from acquired resistance, and effects that are always found in the same isolates cannot be distinguished. While it may be possible to address these limitations we do not attempt to do so here. There are simply too many interactions (over 200 billion) among the full set of variants for our current implementation. After these reductions we have a more tractable $259 \times 75,715$ entry matrix, sufficiently small that all approx. 5.7 billion effects and interactions can be processed using under 250GB of memory.

2.6.3. *Data Sources. P. aeruginosa*genome sequences were selected from strains whose MIC values (Ciprofloxacin) were known. 167 genomes were sourced from the publicly available IPCD International Pseudomonas Consortium Database [18] and 92 genomes were from QIMR Brisbane Australia [19]. The IPCD data consisted of 2 x 300 bp MiSeq reads whilst the QIMR data was 2 x 150 bp reads. The MIC values were obtained as a combination of e-test strips [30] and plate-based assays [31].

3. Results

In this section, we summarise the results of a simulation study we carried out to compare our method against existing approaches. We also demonstrate our method on two large-scale experimental data sets. Note that both sets are too large to attempt using known approaches such as `glinternet` for comparison. In both cases the true interactions are unknown, making the true accuracy of our method in these cases difficult to determine. We nonetheless include these as reasonable examples of cases in which our method is applicable and validate the results by comparing them with known protein interactions [34].

3.1. **Simulation Performance.** Our method aims to have comparable precision and recall to the best performing approach in our previous work [12] while scaling to much larger data sets. To evaluate the accuracy of our method, we compare precision and recall of our method with `glinternet`, the most accurate of the methods tested [12].

Since we achieved the best results only using `glinternet` for variable-selection, then fitting the non-zero beta values with ordinary least squares (OLS) regression, we do the same here. We use `Pint` in the same way and restrict to the first $2,000$ non-zero beta values, rather than using adaptive calibration, which returns too few columns for the OLS regression step.

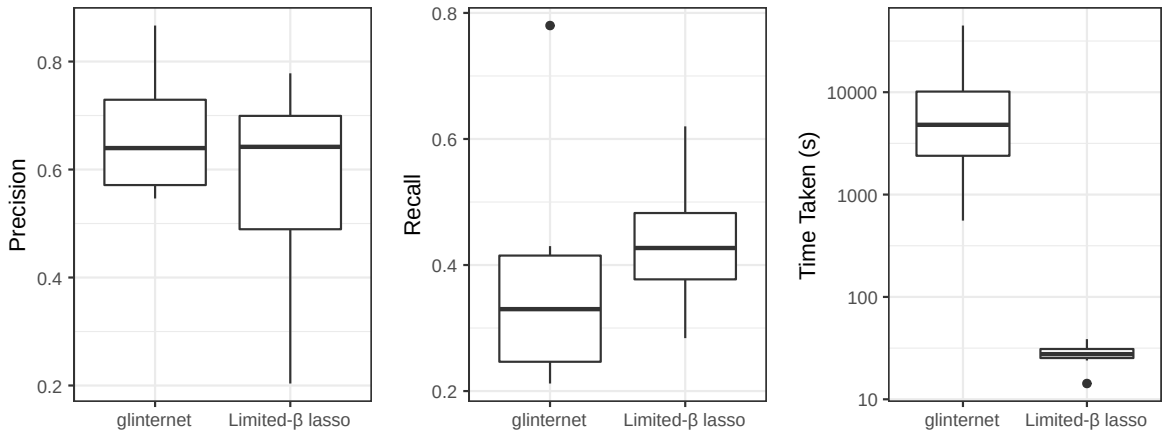
TABLE 1. Runtime comparison between our method and `glinternet`.

X matrix size	`glinternet` time (s)	`Pint` time (s)
$n = 100,\ p = 1,000$	178	2.00
$n = 1,000,\ p = 10,000$	4807	27.6

TABLE 2. Infectx proposed interactions

Gene Names		Estimated Effect	p-value
TTN	KMT2D	0.085	4.35e-05
TTN	PLEC	0.053	1.95e-2
TTN	TTC7B	0.068	2.01e-3
TTN	OBSCN	0.137	8.00e-11
TTN	CDH23	-0.022	1.00e-1

Testing with the same data as in [12], our method is able to identify significantly more correct interactions than `glinternet` (Fig. 5). Precision is largely comparable, with a few outliers in which we see significantly more false positives with our method (Fig. 5a). The run time is orders of magnitude faster than `glinternet`, typically taking 20 to 30 seconds rather than several hours (Table 1 and Fig. 5c). To test the scalability of our implementation, we also run it with the same $2,000$ effect limit on a much larger data set. With $p \approx 27,000, n \approx 30,000$, using 16 SMT threads on a single eight core CPU, we propose 97 main effects and 236 interactions in one and a half hours.



(a) (b) (c)

FIGURE 5. Searching for interactions with `glinternet` vs. our shuffled compressed lasso, using $p = 1,000, n = 10,000$ data from [12]. (a) Precision. (b) Recall. (c) Time taken (log scale).

3.2. **InfectX siRNA Data.** We run our lasso model on the InfectX data (Section 2.6.1) allowing all pairwise interactions, and halting at $\lambda = 0.05$ or the first $2,000$ non-zero effects, whichever comes first. Only the genes and gene-pairs with non-zero predicted effects are then included in the matrix $\mathbf{Z}$. Last, we fit the phenotype $\mathbf{Y}$ to this matrix using least-squares regression $\mathbf{Y} \sim \mathbf{Z}$, using these unbiased estimates and p-values as our final result.

We find 26 proposed effects (21 main and 5 interactions) in under two hours. Our method proposes interactions between five genes and TTN, with varying estimated strengths (see Table 2). Two of these interactions, OBSCN and PLEC, are known protein interactions [34].

We find the same set of interactions in repeated runs (bearing in mind that the matrix is shuffled differently each time). This suggests that these are not random choices, but effects strongly supported by the data. The Adjusted R^2 value is only ≈ 0.088, however, indicating

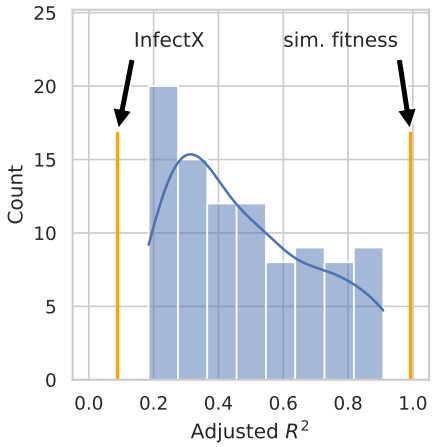


FIGURE 6. Adjusted R^2 density of simulations in Section 3.1, with additional lines indicating the values using InfectX data, or InfectX with simulated fitness, instead.

that while a better than random fit has been found, the chosen effects do not explain the overall observed fitness particularly well. To investigate this, we consider the difference between fitting two different phenotypes. Firstly, the predicted effects with the measured cell counts from InfectX, and secondly a simulated set that reflects our assumptions.

For the simulation we used the same $\mathbf{X}$ matrix, but simulated the fitness effects $\mathbf{Y}$ as a linear combination of randomly chosen gene effects and gene-pair interactions. Every gene had a 10% chance of being assigned an effect, which were sampled from $\mathcal{N}(0, 2)$. We gave every pair of genes a 0.1% chance of an effect, which were also sampled from $\mathcal{N}(0, 2)$. For every row i of $\mathbf{X}$, the fitness value y_i is the sum of both main and interaction effects present, with additional random noise.

$$y_i = \sum_{j=1}^{p} \left(X_{i,j} \, \texttt{effect}(j) + \sum_{k=j+1}^{p} (\texttt{effect}(j,k)) \right) + \mathcal{N}(0, 10)$$

With this simulated phenotype vector, re-running the interaction search with the same parameters, we have an R^2 of ≈ 0.99.

After adjusting for the number of effects proposed, we find that while the fit is better than random using the Z-scored InfectX cell-count as phenotypes, it is not nearly as good as in our simulations. This suggests that at least some of our assumptions are incorrect, namely that our fitness proxy (log cell count) is additive and can be largely explained with individual and pairwise silencing effects, and that the off-target predictions are accurate. While all of these assumptions are somewhat suspect, it should be noted that our siRNA off-target predictions likely miss a significant number of strong effects, and include genes that are not completely silenced [1]. With this in mind, it is plausible that even if the cell count responds to gene silencing according to our assumptions, the predicted effects may not be significantly better than random until accurate siRNA off-target predictions are available.

3.3. **Antibacterial Resistance.** We fit our antibacterial resistance data (see Section 2.6.2) with three different sets of parameters. First, we allow all interactions. Second, we restrict to interactions within 100 columns of each other. Finally, we restrict to interactions within 10 columns of each other. In all cases, we run until the adaptive calibration stopping condition is met. In the first case, allowing all interactions, we find that repeated runs do not suggest any of the same effects. Since the data only contains 259 samples for over 75,000 effects (and over 2.5 billion interactions) it is unsurprising that there are several equally good solutions. We fail to find a reproducible result here because the data simply does not suggest one, and this run is included only to demonstrate that our implementation works at this scale. The second

and third cases produce more consistent results, with some common interactions suggested in both cases. While limiting to interactions within 100 columns rules out the majority of possible interactions, it also limits the number possible solutions enough that we can find one reliably with only 259 samples.

Moreover, when we reduce the interaction matrix to only the non-zero predicted effects, and produce an unbiased fit with least-squares regression, we find that our fit explains the variance in resistance extremely well. Restricting interactions to effects within 100 entries of each other, we have a multiple R^2 of 0.99, and an adjusted R^2 of 0.86. Even limiting to interactions within ten entries, we have a multiple R^2 of 0.78, and an adjusted R^2 of 0.63. These suggest that in this case our model is a particularly good fit.

There were 16 sets of variants found in both limited-distance runs (interactions within 10 or 100 columns only). For each of the 16 SNVs their genes, functions, and interactions were assessed. Genes were identified based on PAO1 reference co-ordinates using Artemis [7]. The STRING[34] database was used to assess the validity of the protein-protein interactions.

Five of the SNV pairs occurred in the same gene. There were four pairs that had high interaction scores > 0.7 and two of pairs were identified twice. Many SNV's were found in genes that encoded for proteins involved in biofilm formation and maintenance indicative of long term chronic infections that are often associated with general antibiotic resistance. Other than pilY1, no other gene was found to be mutated in the lab-based evolution study [31].

There were two pairwise effects that had significant p-values in both runs. The first of these pairs occurred in a gene that encoded a copper resistance protein. The second pair was found in a gene that encodes an RNA binding methyltransferase.

4. Discussion

Genotype-phenotype data sets have recently become available at a never before seen scale. In principle, it is possible to infer not only the effect of individual genomic variants within such data, but of pairwise combinations of their effects. While this has been shown to work in theory, and a number of tools have been developed that work on a smaller scale, there is a shortage of effective methods for human genome-scale data. In this paper we present a regression based method for such large-scale inference of pairwise effects.

Our method performs coordinate descent lasso-regression on a matrix containing all pairwise interactions present in the data. For such an approach to work at scale, we had to make a number of improvements. First we parallelised the algorithm by dividing the matrix into shuffled sets for each thread. We then drastically increased the scale of tractable data sets by compressing columns of the matrix using Simple-8b. Combined with the typically sparse binary nature of genotype-phenotype screens, our method is able to effectively consider hundreds of millions of possible interactions.

We compared the accuracy and running time of our work to `glinternet`, the best of the methods we used previously [12], and found that our method provides comparable accuracy and precision while running hundreds of times faster. We also tested our method using two genome-scale real data sets. One is an exome-wide siRNA perturbation screen ($n \approxeq 67,000$ siRNAs and $p \approxeq 19,000$ genes). The other measures antibacterial resistance with respect to genetic variations in *Pseudomonas aeruginosa*, and includes over two billion possible pairwise interactions. In both cases our method finds a number of effects that are either plausible or previously known.

In some cases we can significantly improve the running time and memory use by only considering local interactions. If interactions are restricted to those within $1,000$ positions of each other, we can search our siRNA screen using $\approx$ 40GB of memory in $\approx$ 20 minutes.

While our method is effective on this scale, there are some limitations that would make it difficult to use on significantly larger data sets. Both the time and space requirements are quadratic in the input sequence, and performance does not scale well with non-uniform memory access. This essentially limits our approach to data that fits in memory on a single machine. The pairwise additive model is also something of an oversimplification. It remains unclear to

what extent genetic effects be treated as additive, and ignoring interactions among of more than two items could well be leaving out the most important effects. In this case we may end up spuriously associating phenotype changes with individual and pairwise effects that just happen to be present, rather than the true, more complicated, interaction.

There are nonetheless a number of opportunities to expand upon this work. If the original $\mathbf{X}$ matrix is sparse, and the pairwise interaction matrix $\mathbf{X}_2$ is very sparse, we would expect three-way interaction columns of an $\mathbf{X}_3$ matrix to be even more so. If there are few enough non-zeros in such a matrix, it may be possible to extend our method beyond pairwise interactions without any fundamental changes. While there would be p^3 columns in a three-way interaction matrix, if the vast majority contain only zeros we may still be able to store it. The indices of non-zero three-way interaction columns could themselves be stored in a compressed list of offsets. Any column whose index is not in this list could then be presumed to be zero and left out of beta updates. Since the memory and time requirements only grow with the number of non-zero entries, this could provide a well be enough for sufficiently sparse data.

Alternatively, as we showed in Section 2.5, we can significantly increase the scale of interaction inference methods by reducing the search space. A more targeted approach than restricting the genome distance, estimating distance in 3D space using Hi-C [4] for example, would drastically reduce the time and space requirements, allowing higher order interactions to be considered.

Finally, the interactions proposed in Section 3.2 that have not already been confirmed may well be real, and are worth further investigation.

Our method is implemented in C, and an R package is provided at github.com/bioDS/pint.

References

[1] Ferhat Alkan et al. “RIsearch2: Suffix Array-Based Large-Scale Prediction of RNA–RNA Interactions and siRNA off-Targets”. In: *Nucleic Acids Research* 45.8 (May 5, 2017), e60–e60. ISSN: 0305-1048. DOI: 10.1093/nar/gkw1325. URL: https://academic.oup.com/nar/article/45/8/e60/2929519 (visited on 11/17/2020).

[2] Vo Ngoc Anh and Alistair Moffat. “Inverted Index Compression Using Word-Aligned Binary Codes”. In: *Information Retrieval* 8.1 (Jan. 1, 2005), pp. 151–166. ISSN: 1573-7659. DOI: 10.1023/B:INRT.0000048490.99518.5c. URL: https://doi.org/10.1023/B:INRT.0000048490.99518.5c (visited on 08/30/2020).

[3] Alan Ashworth, Christopher J. Lord, and Jorge S. Reis-Filho. “Genetic Interactions in Cancer Progression and Treatment”. In: *Cell* 145.1 (Apr. 1, 2011), pp. 30–38. ISSN: 0092-8674. DOI: 10.1016/j.cell.2011.03.020. URL: http://www.sciencedirect.com/science/article/pii/S0092867411002972 (visited on 06/02/2020).

[4] Jon-Matthew Belton et al. “Hi-C: A Comprehensive Technique to Capture the Conformation of Genomes”. In: *Methods (San Diego, Calif.)* 58.3 (Nov. 2012), pp. 268–276. ISSN: 1095-9130. DOI: 10.1016/j.ymeth.2012.05.001. pmid: 22652625.

[5] João Botelho, Filipa Grosso, and Luísa Peixe. “Antibiotic Resistance in Pseudomonas Aeruginosa – Mechanisms, Epidemiology and Evolution”. In: *Drug Resistance Updates* 44 (May 1, 2019), p. 100640. ISSN: 1368-7646. DOI: 10.1016/j.drup.2019.07.002. URL: http://www.sciencedirect.com/science/article/pii/S1368764619300238 (visited on 01/25/2021).

[6] Joseph K. Bradley et al. *Parallel Coordinate Descent for L1-Regularized Loss Minimization*. May 26, 2011. arXiv: 1105.5379 [cs, math]. URL: http://arxiv.org/abs/1105.5379 (visited on 07/14/2019).

[7] T. Carver et al. “Artemis: An Integrated Platform for Visualization and Analysis of High-Throughput Sequence-Based Experimental Data”. In: *Bioinformatics* 28.4 (Feb. 15, 2012), pp. 464–469. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btr703. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr703 (visited on 01/07/2021).

[8] Michael Chichignoud, Johannes Lederer, and Martin J Wainwright. “A Practical Scheme and Fast Algorithm to Tune the Lasso With Optimality Guarantees”. In: (), p. 20.

[9] M. Clamp et al. "Distinguishing Protein-Coding and Noncoding Genes in the Human Genome". In: *Proceedings of the National Academy of Sciences* 104.49 (Dec. 4, 2007), pp. 19428–19433. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0709013104. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.0709013104 (visited on 08/31/2020).

[10] Michael Costanzo et al. "The Genetic Landscape of a Cell." In: *Science* (2010).

[11] Richard Durstenfeld. "Algorithm 235: Random Permutation". In: *Communications of the ACM* 7.7 (July 1964), p. 420. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/364520.364540. URL: https://dl.acm.org/doi/10.1145/364520.364540 (visited on 10/28/2020).

[12] Kieran Elmes et al. "Learning Epistatic Gene Interactions from Perturbation Screens". In: *bioRxiv* (Aug. 25, 2020), p. 2020.08.24.264713. DOI: 10.1101/2020.08.24.264713. URL: https://www.biorxiv.org/content/10.1101/2020.08.24.264713v1 (visited on 08/31/2020).

[13] Ronald A Fisher and Frank Yates. *Statistical Tables: For Biological, Agricultural and Medical Research*. Oliver and Boyd, 1938.

[14] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010). ISSN: 1548-7660. DOI: 10.18637/jss.v033.i01. pmid: 20808728. URL: http://www.jstatsoft.org/v33/i01/ (visited on 07/14/2019).

[15] Wenjiang J. Fu. "Penalized Regressions: The Bridge versus the Lasso". In: *Journal of Computational and Graphical Statistics* 7.3 (Sept. 1998), pp. 397–416. ISSN: 1061-8600, 1537-2715. DOI: 10.1080/10618600.1998.10474784. URL: http://www.tandfonline.com/doi/abs/10.1080/10618600.1998.10474784 (visited on 06/19/2020).

[16] Robert Gaynes, Jonathan R. Edwards, and National Nosocomial Infections Surveillance System. "Overview of Nosocomial Infections Caused by Gram-Negative Bacilli". In: *Clinical Infectious Diseases* 41.6 (Sept. 15, 2005), pp. 848–854. ISSN: 1058-4838. DOI: 10.1086/432803. URL: https://doi.org/10.1086/432803 (visited on 01/25/2021).

[17] *GRCh38.P13 - Genome - Assembly - NCBI*. URL: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.39 (visited on 12/01/2020).

[18] *IPCD International Pseudomonas Consortium Database*. URL: https://ipcd.ibis.ulaval.ca/ (visited on 01/07/2021).

[19] Timothy J. Kidd et al. "Pseudomonas Aeruginosa Exhibits Frequent Recombination, but Only a Limited Association between Genotype and Ecological Setting". In: *PLoS ONE* 7.9 (Sept. 6, 2012). Ed. by Sam Paul Brown, e44199. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0044199. URL: https://dx.plos.org/10.1371/journal.pone.0044199 (visited on 01/07/2021).

[20] Ben Langmead and Steven L. Salzberg. "Fast Gapped-Read Alignment with Bowtie 2". In: *Nature Methods* 9.4 (4 Apr. 2012), pp. 357–359. ISSN: 1548-7105. DOI: 10.1038/nmeth.1923. URL: https://www.nature.com/articles/nmeth.1923 (visited on 01/10/2021).

[21] D. Lemire and L. Boytsov. "Decoding Billions of Integers per Second through Vectorization". In: *Software: Practice and Experience* 45.1 (2015), pp. 1–29. ISSN: 1097-024X. DOI: 10.1002/spe.2203. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2203 (visited on 07/13/2020).

[22] Jeffrey B Lyczak, Carolyn L Cannon, and Gerald B Pier. "Establishment of Pseudomonas Aeruginosa Infection: Lessons from a Versatile Opportunist1*Address for Correspondence: Channing Laboratory, 181 Longwood Avenue, Boston, MA 02115, USA". In: *Microbes and Infection* 2.9 (July 1, 2000), pp. 1051–1060. ISSN: 1286-4579. DOI: 10.1016/S1286-4579(00)01259-4. URL: http://www.sciencedirect.com/science/article/pii/S1286457900012594 (visited on 01/25/2021).

[23] Antonio Mallia, Michał Siedlaczek, and Torsten Suel. "An Experimental Study of Index Compression and DAAT Query Processing Methods". In: *Advances in Information Retrieval*. Ed. by Leif Azzopardi et al. Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 353–368. ISBN: 978-3-030-15712-8.

[24] Ramachandra Nanjegowda et al. "Scalability Evaluation of Barrier Algorithms for OpenMP". In: *Evolving OpenMP in an Age of Extreme Parallelism*. Ed. by Matthias S. Müller, Bronis

R. de Supinski, and Barbara M. Chapman. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 42–52. ISBN: 978-3-642-02303-3. DOI: 10.1007/978-3-642-02303-3_4.

[25] Preeti Pachori, Ragini Gothalwal, and Puneet Gandhi. "Emergence of Antibiotic Resistance Pseudomonas Aeruginosa in Intensive Care Unit; a Critical Review". In: *Genes & Diseases* 6.2 (June 1, 2019), pp. 109–119. ISSN: 2352-3042. DOI: 10.1016/j.gendis.2019.04.001. URL: http://www.sciencedirect.com/science/article/pii/S2352304219300170 (visited on 01/25/2021).

[26] Zheng Pang et al. "Antibiotic Resistance in Pseudomonas Aeruginosa: Mechanisms and Alternative Therapeutic Strategies". In: *Biotechnology Advances* 37.1 (Jan. 1, 2019), pp. 177–192. ISSN: 0734-9750. DOI: 10.1016/j.biotechadv.2018.11.013. URL: http://www.sciencedirect.com/science/article/pii/S0734975018301976 (visited on 01/25/2021).

[27] powturbo. *Powturbo/TurboPFor-Integer-Compression*. July 9, 2020. URL: https://github.com/powturbo/TurboPFor-Integer-Compression (visited on 07/09/2020).

[28] Olga Puchta et al. "Network of Epistatic Interactions within a Yeast snoRNA". In: *Science* 352.6287 (May 13, 2016), pp. 840–844. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aaf0965. pmid: 27080103. URL: https://science.sciencemag.org/content/352/6287/840 (visited on 08/13/2020).

[29] Pauli Rämö et al. "Simultaneous Analysis of Large-Scale RNAi Screens for Pathogen Entry". In: *BMC Genomics* 15.1 (Dec. 22, 2014), p. 1162. ISSN: 1471-2164. DOI: 10.1186/1471-2164-15-1162. URL: https://doi.org/10.1186/1471-2164-15-1162 (visited on 11/14/2019).

[30] Kay A. Ramsay et al. "Genomic and Phenotypic Comparison of Environmental and Patient-Derived Isolates of Pseudomonas Aeruginosa Suggest That Antimicrobial Resistance Is Rare within the Environment". In: *Journal of Medical Microbiology* 68.11 (Nov. 1, 2019), pp. 1591–1595. ISSN: 0022-2615, 1473-5644. DOI: 10.1099/jmm.0.001085. URL: https://www.microbiologyresearch.org/content/journal/jmm/10.1099/jmm.0.001085 (visited on 01/07/2021).

[31] Attika Rehman, Wayne M. Patrick, and Iain L. Lamont. "Mechanisms of Ciprofloxacin Resistance in Pseudomonas Aeruginosa: New Approaches to an Old Problem". In: *Journal of Medical Microbiology,* 68.1 (2019), pp. 1–10. ISSN: 0022-2615, DOI: 10.1099/jmm.0.000873. URL: https://www.microbiologyresearch.org/content/journal/jmm/10.1099/jmm.0.000873 (visited on 01/07/2021).

[32] Tracey Remmington, Nikki Jahnke, and Christian Harkensee. "Oral Anti-Pseudomonal Antibiotics for Cystic Fibrosis". In: *Cochrane Database of Systematic Reviews* (July 14, 2016). Ed. by Cochrane Cystic Fibrosis and Genetic Disorders Group. ISSN: 14651858. DOI: 10.1002/14651858.CD005405.pub4. URL: http://doi.wiley.com/10.1002/14651858.CD005405.pub4 (visited on 01/25/2021).

[33] Benjamin Schlegel, Rainer Gemulla, and Wolfgang Lehner. "Fast Integer Compression Using SIMD Instructions". In: *Proceedings of the Sixth International Workshop on Data Management on New Hardware - DaMoN '10*. The Sixth International Workshop. Indianapolis, Indiana: ACM Press, 2010, pp. 34–40. ISBN: 978-1-4503-0189-3. DOI: 10.1145/1869389.1869394. URL: http://portal.acm.org/citation.cfm?doid=1869389.1869394 (visited on 07/13/2020).

[34] *STRING: Functional Protein Association Networks*. URL: https://string-db.org/cgi/about.pl (visited on 07/22/2020).

[35] Robert Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 2517-6161. DOI: 10.1111/j.2517-6161.1996.tb02080.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x (visited on 08/29/2020).

[36] Andrew Trotman and Jimmy Lin. "In Vacuo and In Situ Evaluation of SIMD Codecs". In: *Proceedings of the 21st Australasian Document Computing Symposium*. ADCS '16. Caulfield, VIC, Australia: Association for Computing Machinery, Dec. 5, 2016, pp. 1–8.

ISBN: 978-1-4503-4865-2. DOI: 10.1145/3015022.3015023. URL: https://doi.org/10.1145/3015022.3015023 (visited on 07/08/2020).

[37] Sara Van de Geer. "The Deterministic Lasso". In: 2007.

[38] Tong Tong Wu and Kenneth Lange. "Coordinate Descent Algorithms for Lasso Penalized Regression". In: *The Annals of Applied Statistics* 2.1 (Mar. 2008), pp. 224–244. ISSN: 1932-6157. DOI: 10.1214/07-AOAS147. arXiv: 0803.3876. URL: http://arxiv.org/abs/0803.3876 (visited on 07/14/2019).