A FAST LASSO-BASED METHOD FOR INFERRING HIGHER-ORDER INTERACTIONS

KIERAN ELMES[1], ASTRA HEYWOOD[2], ZHIYI HUANG[1], AND ALEX GAVRYUSHKIN[1,3] ✉

Abstract. Large-scale genotype-phenotype screens provide a wealth of data for identifying molecular alterations associated with a phenotype. Epistatic effects play an important role in such association studies. For example, siRNA perturbation screens can be used to identify combinatorial 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. Recently developed tools scale to human exome-wide screens for pairwise interactions, but none to date have included the possibility of three-way interactions. Expanding upon recent state-of-the art methods, we make a number of improvements to the performance on large-scale data, making consideration of three-way interactions possible. We demonstrate our proposed method, `Pint`, on both simulated and real data sets, including antibiotic resistance testing and siRNA perturbation screens. `Pint` outperforms known methods in simulated data, and identifies a number of biologically plausible gene effects in both the antibiotic and siRNA models. For example, we have identified a combination of known tumor suppressor genes that is predicted (using `Pint`) to cause a significant increase in cell proliferation.

1. Introduction

Epistatic gene interactions have practical implications for personalised medicine, and synthetic lethal interactions in particular can be used in cancer treatment [4]. Discovering these interactions is currently challenging at a practical scale [34, 24, 13, 15], however. In particular, there are no methods able to infer three-way effects. 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 [12]. This approach does not scale to the approximately 200 million pairwise combinations possible among human protein coding genes, however, let alone 1.3 trillion three-way combinations. We instead consider inferring interactions from large-scale genotype-phenotype data. These include mass knockdown screens, in which a large number of genes are simultaneously suppressed, and the resulting phenotypic effects are measured.

We have shown in [15] that a lasso-based approach to inferring interactions from an siRNA perturbation matrix is a feasible method for large-scale pairwise interaction detection. In this additive model, we assume fitness is a linear combination of the effects of each gene, and the effect of every combination of these genes. For the sake of scalability, in [15] we considered only individual and pairwise effects, and assumed gene suppression was strictly binary. The fitness difference (compared to no knockdowns) in each experiment is then the sum of individual and pairwise effects $\sum_i^p \beta_i g_i + \sum_i^p \sum_{j>i}^n \beta_{i,j}(g_i \cdot g_j)$, 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.

[1]Department of Computer Science, University of Otago, New Zealand
[2]Department of Biochemistry, University of Otago, New Zealand
[3]School of Mathematics and Statistics, University of Canterbury, New Zeleand
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. Furthermore, we aim to go beyond pairwise interactions to consider three-way effects.

A recently developed method, WHInter [38], is effective at solving lasso regression on much larger scale data than `glinternet`. This performance comes as a result of pruning the interactions considered based on the current regularisation parameter, removing interactions that could not possibly have a non-zero effect. Because doing so does not affect the solution of the regression problem, we expect comparable accuracy from WHInter and `glinternet`. Nonetheless there are a number of areas in which we can improve upon WHInter. In particular WHInter does not make use of multi-core CPUs, and considers only pairwise interactions.

We have developed an R-package, `Pint`, that is able to perform square-root lasso regression on all pairwise interactions on a one thousand gene screen, using ten thousand siRNAs, in 15 seconds, and all three-way interactions on the same set in under ten minutes. We are also able to find the largest $2,000$ effects, excluding three-way combinations, on a genome-scale data set with $19,000$ genes and $67,000$ siRNAs in half an hour using two eight-core CPUs. This is made possible by taking into account that our input matrix X is both sparse and strictly binary, parallelising the pruning method from [38], and compressing the active set. To allow three-way interactions, we extend to a two-step pruning method able to rule out both pairwise and three-way interactions. Our package, `Pint`, is available at github.com/bioDS/pint.

Our method is based on an existing fast algorithm [59], which we adapt for use on binary matrices. We further add parallelised version of the pruning step from [38], expanded to include three-way interactions. We provide a detailed description of our implementation, followed by the scalability analysis, below. We also perform a simulation study to compare our method's scalability with known methods, and analyze two large-scale experimental data sets.

In the first, an siRNA perturbation screen from [45], we search for both individual genes and combinations (pairwise or three-way) that have an effect when simultaneously silenced, stopping after the first 131 effects have been identified. The results include 22 individual, 41 pairwise and 68 three-way effects. We investigate the biological plausibility of the top five effects, and find that three out of five are suppressing genes that could be related to cell survival. One combination in particular involves simultaneously disabling two tumor suppressing genes, and is predicted to cause a significant increase in cell proliferation.

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 [22, 36]. *P. aeruginosa* is known to acquire adaptive antibiotic resistance in response to long term usage of antibiotics associated with chronic infections [10, 42, 43]. 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 bactericidal antibiotics that targets DNA replication and is one of the most widely used antibiotics against *P. aeruginosa* [48].

This set contains over $170,000$ SNVs, too many for our method to include all possible three-way interactions. Three-way interactions can be included if we remove columns with less than 30 entries, reducing the matrix to $\approx 76,000$ columns. This ignores over half of the SNVs present however, so we instead limit the search to individual and pairwise combinations of variants, and determine the first 50 effects that are discovered. Among these, 13 of the 16 non-synonymous variants were identified as having possible contributions to Ciprofloxacin resistance.

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

2. Materials and Methods

Throughout the paper we refer to fitness landscapes, and focus on fitness values as our phenotype of interest, but would like to note that the theory can be applied for any (binary) genotype-phenotype mapping and any phenotype. Let a fitness landscape be a mapping $f : \mathcal{P} \to \mathbb{R}$ from the genotype space to fitness values. Furthermore, suppose genotypes are strictly binary, $\mathcal{P} \in \{0, 1\}^p$, where 1 indicates the presence of a particular mutation (or variant), 0 indicates its absence, and p is the number of genes. The complete fitness landscape then describes the effects of all combinations of mutations [5]. For example, the two gene space $\mathcal{P} = \{0,\, 1\}^2$ contains the wild-type 00, two single mutants 01 and 10, and the double mutant 11. The fitness landscape $f : \{0, 1\}^2 \to \mathbb{R}$ in this case can be written as

$$\begin{aligned} f(0,0) &= \beta_0 \\ f(1,0) &= \beta_0 + \beta_1 \\ f(0,1) &= \beta_0 + \beta_2 \\ f(1,1) &= \beta_0 + \beta_1 + \beta_2 + \beta_{1,2} \end{aligned}$$

for parameters $\beta_i \in \mathbb{R}$. β_0 is called the bias, β_1 and β_2 main effects, and $\beta_{1,2}$ the pairwise interaction. This pairwise interaction is exactly the classic definition of epistasis in quantitative genetics [16]. In generalising to higher-order interactions, we follow the definitions of Otwinowski and Nemenman [41]. For $p \geq 3$ genes, the complete fitness landscape f is:

$$f(x_1,\, \ldots,\, x_p) = \beta_0 + \sum_i x_i\beta_i + \sum_{i<j} x_i x_j \beta_{i,j} + \sum_{i<j<k} x_i x_j x_k \beta_{i,j,k} + \ldots \tag{1}$$

While including all possible interactions quickly becomes computationally and statistically intractable for large p, we can model the interactions up to a point. Ignoring interactions of order four and higher we obtain an approximation of the fitness landscape:

$$f(x_1,\, \ldots,\, x_p) \approx \beta_0 + \sum_i x_i\beta_i + \sum_{i<j} x_i x_j \beta_{i,j} + \sum_{i<j<k} x_i x_j x_k \beta_{i,j,k} \tag{2}$$

The remainder of this section describes the regression model we use to estimate these effects, the algorithms we use to efficiently solve it, and finally the data sets on which we apply it.

2.1. Regression Model.

Given as input a matrix $\mathbf{X} \in \{0, 1\}^p$ and a vector $\mathbf{Y} \in \mathbf{R}^n$, where columns of $\mathbf{X}$ are genes or variants of interest, rows are samples from the genotype space, and entries y_i of $\mathbf{Y}$ are the fitness values corresponding corresponding to the ith row of $\mathbf{X}$, our goal is to estimate the parameters $\beta_0, \beta_i, \beta_{i,j}, \beta_{i,j,k}$ of the fitness landscape model in Equation (2), such that for any row x_i of $\mathbf{X}$, $f(x_i) \approxeq y_i$.

To do this we construct a matrix $\mathbf{X}^* \in \{0, 1\}^{n \times p_{int}}$, where $p_{int} = \binom{p}{1} + \binom{p}{2} + \binom{p}{3}$, containing a column for each gene, pair of genes, and triplet of genes. Specifically, to construct $\mathbf{X}^*$ we extend $\mathbf{X}$ by adding the following columns. For every pair of columns i, j and triplet i, j, k we add an interaction columns $X_{i,j,k}$ by taking the element-wise product of the columns X_i and X_j, or X_i, X_j, and X_k (Formula 1).

$$\underset{x_a}{\begin{bmatrix}1\\0\\1\\0\\1\end{bmatrix}} \circ \underset{x_b}{\begin{bmatrix}1\\1\\0\\0\\1\end{bmatrix}} \circ \underset{x_c}{\begin{bmatrix}1\\0\\0\\1\\1\end{bmatrix}} = \underset{x_{a,b,c}}{\begin{bmatrix}1\\0\\0\\0\\1\end{bmatrix}}$$

Formula 1. Creation of three-way interaction effect columns

In cases where we exclude three-way interactions, considering only pairwise interactions and the original columns of $\mathbf{X}$, we call the pairwise interaction matrix $\mathbf{X}_2^*$. Note that we use bold

$\mathbf{X}_2^*$ to refer to this matrix, and non-bold X_2^* to refer to column 2 of the matrix $\mathbf{X}^*$. Given the interaction matrix $\mathbf{X}^* \in \{0,1\}^{n \times p_{int}}$ we estimate effects by solving the square-root lasso, as defined in [55], by minimising the difference between the predicted and actual fitness values, subject to regularisation.

$$\hat{\beta} = \underset{\beta \in \mathbf{R}^{p_{int}}}{\arg\min} \|y - \mathbf{X}^* \beta\|_2 + \lambda \|\beta\|_1$$

2.2. **Cyclic Linear Regression.** Our approach to lasso regression is based on a cyclic coordinate descent algorithm from [20], as described in [59]. This method begins with $\beta_j = 0$ for all j and updates the beta values sequentially, with each update attempting to minimize the current total error. Here this total error is the difference between the effects we have estimated and the fitness we observe. 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 $\mathbf{X}^*$ at column j of row i, the error is the following.

$$\sum_{i=1}^{n} |y_i - \sum_{j=1}^{p_{int}} x_{ij} \cdot \beta_j| \tag{3}$$

The error affected by a single beta value can then be minimized, using lasso regularisation, by updating β_k as follows:

$$\beta_k \leftarrow \begin{cases} max(0, \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{|X_k|} - \lambda) & \text{for } \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{|X_k|} > 0 \\ min(0, \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{|X_k|} + \lambda) & \text{for } \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{|X_k|} < 0 \end{cases} \tag{4}$$

We adjust this to instead solve the square-root lasso ([6]) using Equation (5).

$$\beta_k \leftarrow \begin{cases} max(0, \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{E_{mse}} - \lambda) & \text{for } \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{E_{mse}} > 0 \\ min(0, \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{E_{mse}} + \lambda) & \text{for } \beta_k + \frac{\sum_{i=1}^{n}(x_{ik}(y_i - r_i))}{E_{mse}} < 0 \end{cases} \tag{5}$$

where

$$E_{mse} = \|Y - \mathbf{X}\beta\|_2$$

We cyclically update each β_k until the solution converges for a particular λ, reduce the value of λ, and repeat. To reach the genome-scale we avoid unnecessarily considering most interactions (Section 2.4), compress the matrix (Section 2.5), and parallelise the process (Section 2.6).

2.3. **Choosing Lambda.** The lasso penalty requires a regularisation parameter λ, which effectively decides how large an effect has to be before it will be included in the model. This can range from allowing all values ($\lambda = 0$) to allowing none ($\lambda \to \infty$). Choosing the correct value of λ is essential if we want to include only the significant effects and avoid over-fitting noise. For the standard lasso this is typically done by choosing an initial value sufficiently large that all beta values will be zero and gradually reducing λ, fitting the model for each value until a stopping point chosen with K-fold cross-validation [19]. Cross-validation requires fitting each λ value K times, however, significantly increasing the running time. The square root lasso performs well with an easily calculated λ value, independent of the standard deviation of noise [6]. We use this lower limit of $\lambda = 1.1 \times \frac{1}{\sqrt{n}} \phi^{-1}(\frac{0.95}{2 \times p})$, where ϕ is the probability density function of the standard normal distribution and p is the number of columns of the $\mathbf{X}^*$ matrix. Note that Belloni, Chernozhukov, and Wang [6] actually use the penalty $\frac{\lambda}{n}$, whereas we use λ directly as in Tian, Loftus, and Taylor [55]. To reach the same minimum value as in [6], our equation differs from theirs by a factor of n.

2.4. **Pruning.** We implement the branch pruning (Equation (6)) and working set (Algorithm 1) algorithms from WHInter [38] to avoid considering unnecessary effects at each value of λ.

The pruning algorithm determines whether any interaction with a particular effect i can be non-zero at the current λ, and if not removes all such interactions i, j. Effects that may have a sufficiently large interaction are instead included in the working set. We give a brief overview of this algorithm here, and refer the reader to [38] for further details.

For the square-root lasso penalty we keep track of $\omega = \|r\|_2$ at each iteration, where $r = Y - \mathbf{X}\beta$. For a particular column index x in $\mathbf{X}^*$, we further define:

$$\phi_x = r, \text{the last time the column } x \text{ was included in the working set.}$$
$$\pi_x = \max_{j \in (max(x), p]} (X_x^* \cdot X_j \cdot \phi_x)$$

where x may be a set $\{i, j\}$ for the column $X_x^* = X_i \circ X_j$. The upper bound $\eta(x)$ for any interaction with x is then:

$$\eta(x) = \alpha \cdot \pi_x + \max_{r \in \{r^+, r^-\}} |X_x^* \cdot r - \phi_x \cdot \alpha| \tag{6}$$

where:

$$r_i^+ = \begin{cases} r_i, & \text{if } r_i > 0. \\ 0, & \text{otherwise.} \end{cases}$$
$$r_i^- = \begin{cases} r_i, & \text{if } r_i < 0. \\ 0, & \text{otherwise.} \end{cases}$$
$$\alpha_x = \left| \frac{x \cdot r \cdot \phi_x}{\|\phi_x\|_2^2} \right|$$

According to the Karush-Kuhn-Tucker conditions for the square-root lasso, no effect can have a value less than $|\lambda \cdot \omega|$ [55]. We can therefore ignore any interactions whose effect has an upper bound below $|\lambda \cdot \omega|$. The working set contains the columns that have not been ruled out.

This fast step allows us to rule out most effects without even calculating them. Many interactions even among these columns will still never be updated at the current value of λ however. As in [38] we further reduce the problem by calculating the *active set*, the subset of the working set that will be updated in the current iteration. To do so, we iterate through the entire working set one time, calculating all interactions and updating ϕ and π for each, and adding those that are significant enough to the active set (Algorithms 1 and 3).

2.4.1. *Active-set for Pairwise Interactions.* For each pair i, j of effects that have not been ruled out, we need the sum of their row's residuals $r \cdot X_i \cdot X_j$. As in [38], rather than taking all the columns and reading through both to find the places they overlap we store the matrix in both column and row major versions and read through only the first column $\mathbf{X}_i$. All interactions are found by reading the row k for each non-zero entry in X_i. Since the matrix is stored in a sparse format, we find all pairwise interactions with $\mathbf{X}_i$ in $O(\sum_{j=i+1}^{p} |X_i \circ X_j|)$ operations. This is further improved in our implementation by calculating a reduced row-major version of X, X', containing only the effects present in the working set. Once the active set has been calculated, we solve the regression problem for the current λ with Algorithm 2. In `Pint` we solve the main effects β_i alone first, followed by pairwise effects $\beta_{i,j}$ (and finally three-way effects $\beta_{i,j,k}$). This ensures pairwise effects are only used to explain variance that cannot be fit using main effects. Similarly, three-way effects should only be introduced where pairwise effects are inadequate.

```
W ← ∅;
for i ∈ 1...p do
    for a ∈ 1...n | X'_{a,i} ≠ 0 do
        sum_i ← sum_i + r_a;
        for j ∈ i+1...p | X'_{a,j} ≠ 0 do
            sum_{i,j} ← sum_{i,j} + r_a;
        end
    end
    γ ← |sum_i|;
    if sum_i > λ · ω then
        W ← W ∪ {{i}};
    end
    for j ∈ i+1...p do
        γ ← max(γ, |sum_{i,j}|);
        if |sum_{i,j}| > λ · ω then
            W ← W ∪ {{i, j}};
        end
    end
    π_i ← γ;
    φ_i ← r;
end
return W
```

Algorithm 1: Identify Active Set (WHInter version, pairwise interactions only).

```
while not converged do
    for k ∈ 1...p_int do
        Δβ_k ← (Σ_{i=1}^{n} (x_ik (y_i − r_i))) / E_mse;
        if |β_k + Δβ_k| > λ then
            β'_k ← β_k;
            β_k ← β_k + Δβ_k;
            if β_k > 0 then
                β_k ← β_k − λ;
            end
            else
                if β_k < 0 then
                    β_k ← β_k + λ;
                end
            end
            for i ∈ 1...n do
                r_i ← r_i + x_{i,j} · (β_k − β'_k);
            end
        end
    end
end
```

Algorithm 2: Sequential Cyclic Sub-problem Algorithm.

2.4.2. *Active-set for Three-way Interactions.* WHInter's pruning algorithm (Algorithm 1) can be extended to three way interactions with a second-level pruning step while updating the active set (Algorithm 3). The three-way active set W can then be solved as before using Algorithm 2. Since $\eta(x)$ requires the column X^*_x, to be calculated, and we are now using interaction columns $x = \{i, j\}$, we keep a cache of every interaction $X^*_{i,j}$ calculated so far and re-use them. For

sufficiently small λ this may become the majority of $\mathbf{X}^*$, so we compress these columns using Simple-8b (Section 2.5). The upper bound $\eta(\{i,j\})$ may be re-used and should also be cached.

```
W ← ∅;
for i ∈ 1...p do
    sum_i ← sum_i + r_a;
    for a ∈ 1..n | X'_{a,i} ≠ 0 do
        for j ∈ i+1...p | X'_{a,j} ≠ 0 do
            sum_{i,j} ← sum_{i,j} + r_a;
            if η({i,j}) > λ·ω then
                for k ∈ j+1...p | X'_{a,k} ≠ 0 do
                    sum_{i,j,k} ← sum_{i,j,k} + r_a;
                end
            end
        end
    end
    γ ← |sum_i|;
    if sum_i > λ·ω then
        W ← W ∪ {{i}};
    end
    for j ∈ i+1...p do
        γ ← max(γ, |sum_{i,j}|);
        if sum_{i,j} > λ·ω then
            W ← W ∪ {{i,j}};
        end
        γ* ← 0;
        for k ∈ j+1...p do
            γ* ← max(γ*, |sum_{i,j,k}|);
            if sum_{i,j,k} > λ·ω then
                W ← W ∪ {{i,j,k}};
            end
            π_{i,j} ← γ*;
            φ_{i,j} ← r*;
        end
        γ ← max(γ, γ*);
    end
    π_i ← γ;
    φ_i ← r;
end
return W
```

Algorithm 3: Identify Active Set (`Pint` version, three-way interactions)

2.5. **Compression.** In our method (Algorithm 3), the input matrix $\mathbf{X}$ is accessed frequently, iterating through both rows and columns. Because the matrix is not prohibitively large, we store it as both a column and row-major *uncompressed* sparse matrix. The active set, on the other hand, can be as large as $\mathbf{X}^*$. We considerably increase the number of possible non-zero effects by storing this only as a set of Simple-8b compressed columns (Figure 1 (c)). 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 column. These small integers can then be efficiently compressed with any of a range of integer compression techniques (Figure 1), a subject that has been heavily developed for Information Retrieval. We compare a number of such methods, including the Simple-8b algorithm from [51] (which we implement and use in our package) in Appendix A.

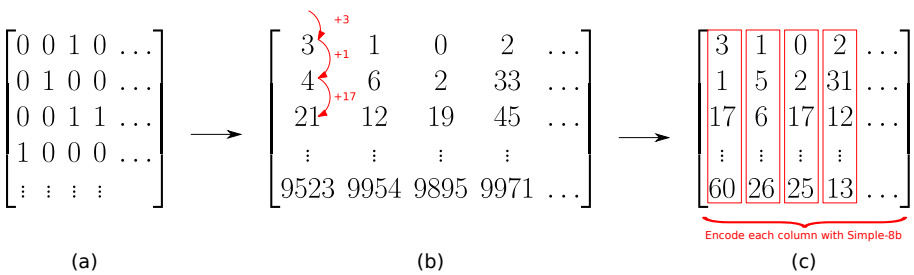


FIGURE 1. Matrix Compression. Given a full matrix (a), we reduce it to the indices of non-zero entries (b), then the compressed difference between these (c). Arrows represent transitions between different representations.

2.6. **Parallelisation.** The three components of the algorithm, pruning, active set calculation, and solving the sub-problem, can all be done in parallel. Pruning and active-set calculation are trivially parallelisable, and performance scales well as long as each thread is given a large separate chunk of work. In practice this means dividing the matrix into several continuous chunks for the pruning step, one for each thread. For the active set calculation we calculate all two and three way interactions with a particular column on the same thread. As well as keeping each threads workload sufficiently large, this means cached interaction columns and upper bounds can be kept thread-local.

The sub-problem (Algorithm 2) can also be parallelised, and performs well when the active set is sufficiently large and shuffled before each iteration [11]. Parallelising updates to a small active set can significantly harm performance however. In practice this presents a number of difficulties. The active set contains only the columns that would be non-zero at the current value of λ, and this is initially high, with the active set containing very few elements. It is therefore not worth parallelising until λ reaches a value where sufficiently many non-zero effects are allowed. In our testing we often reached the final value, $\lambda = 1.1 \times \frac{1}{\sqrt{n}}\phi^{-1}(\frac{0.95}{2\times p})$, before this occurred.

Furthermore, as λ decreases calculating the active set quickly dominates the running time. While not parallelising the sub-problem calculation theoretically limits the best-case performance of our method, it is a minor limitation in practice. We therefore keep this component single-threaded in our implementation in `Pint`.

We demonstrate the parallel scalability using a simulated data set of $n = 8,000$ rows and $p = 4,000$ columns, running `Pint` until the first 500 pairwise or main effects have been found and recording the median running-time out of five runs. As we see in Figure 2, performance scales up to an 8× speed-up using 32 threads on 16 cores across two CPUs, which is typical for a highly parallel task running on CPU(s) with shared memory [31].

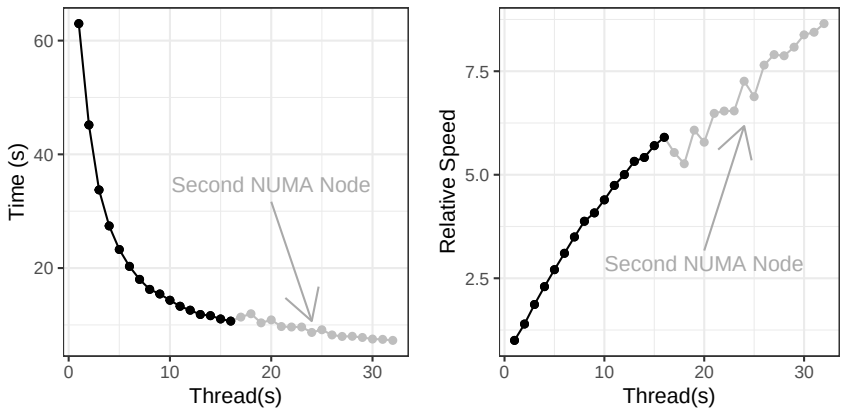


FIGURE 2. Running time on an increasing number of threads. Note that performance initially decreases with only a small number of threads on a second NUMA node. Tests were performed using two Intel Xeon Gold 6244 CPUs

2.7. **Data.** We prepared two experimental data sets to evaluate our method and test the scalability of our implementation. The first is the InfectX siRNA perturbation screen [52] in which siRNAs are applied to an infected human cell line. We predict off-target effects across the entire exome, and use these 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.

2.7.1. *InfectX siRNA Data.* To demonstrate our method on real genome-scale data, we use the mock group from InfectX [45]. This set contains $6,703$ siRNA perturbations (excluding control wells and pooled siRNAs). Off-target effects are prediction using RIsearch2 [2]. 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 [25]). We use an energy cut-off of -20 and match the entire siRNA (rather than only the $3'$ UTR) as suggested in [2].

We then form the $6,703 \times 19,353$ matrix of off-target effects with columns for each gene, and rows for each siRNA as in [15]. 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, Z-scoring then normalises each well's score with respect to the rest of it's plate.

2.7.2. *Antibacterial Resistance.* SNVs from 259 isolates of *Pseudomonas aeruginosa* were sequenced using Illumina technologies (IPCD isolates on MiSeq and QIMR isolates on HISeq). SNVs from raw reads were mapped to the reference genome PAO1 using Bowtie2 (v. 2.3.4) [30] read aligners. Variant reports were then sorted into a table, 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. We removed SNVs that occur less than five times, resulting in a table of 259 rows and $174,334$ columns.

P. aeruginosa genome sequences were selected from strains for which MIC values (Ciprofloxacin) were known. 167 genomes were sourced from the publicly available IPCD International Pseudomonas Consortium Database [26] and 92 genomes were from QIMR Brisbane Australia [29]. 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 [46] and plate-based assays [47].

3. Results

In this section, we summarize 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. We include these as reasonable examples of cases in which our method is applicable and validate the results by comparing them with known effects in the String and NCBI Gene databases [54, 8].

3.1. Simulation Performance.

Our method aims to have comparable precision and recall to the best performing approach known to us [15] while scaling to much larger data sets. To evaluate the accuracy of our method `Pint`, we compare precision and recall with `glinternet`, the most accurate of the methods tested [15], and WHInter, a recent fast method based on the idea of limited working sets [38].

All methods use regularised regression in some form or another, and for a fair comparison we use the same parameters wherever possible. We instruct each method to stop after 200 non-zero effects have been found in the smaller sets and 500 in the larger, assume there is no bias β_0, and use a convergence threshold of 1%. Although we give the same parameters, all methods return more non-zero effects than we request, to varying extents. `glinternet` in particular typically gives 300 non-zero effects instead of 200, and close to $1,000$ instead of 500. To keep the running time manageable we do not use cross-validation in `glinternet`. `Pint` and `glinternet` were run in parallel where possible using 32 SMT threads across 16 cores across two CPUs. Note that a significant portion of `glinternet`'s running time is single-threaded, and further increasing the number of threads has to noticeable impact on performance. WHInter implements a sequential algorithm, so was run on a single thread. To assess performance we compare the predicted non-zero main effects and pairwise interactions, β_i and $\beta_{i,j}$, with the simulated effects, recording precision and recall. We further compare the running time of each method. Three-way effects are excluded since neither `glinternet` nor WHInter are able to produce them.

Testing with the same data as in [15], we find `Pint` and WHInter perform comparably in terms of precision and recall (Figures 3a, 3b, 3d and 3e). This is unsurprising considering both are solving similar regression problems with the same parameters. `glinternet` typically has lower precision and higher recall than the others, which we would expect since it returns significantly more effects than requested. `Pint` is consistently faster than both WHInter and `glinternet` on both small and large data sets (Figures 3c and 3f). In both the small and large test cases, we see an improvement in running time compared to WHInter, the fastest method currently available. Using the larger data sets (Figure 3f) this is because our method runs in parallel, whereas WHInter does not, yielding a significant performance improvement. In the small test cases (Figure 3c) the majority of the used memory fits into the cache on a single core, and we do not see a large improvement from parallelisation. We nonetheless find that `Pint` is nearly three times faster than WHInter, perhaps due to less efficient cache usage in the latter.

3.2. Three-way Effects.

In cases where three-way effects are present, we find that including them in the search improves not only the overall fit, but the accuracy of pairwise predictions. This makes sense when we consider that the alternative is to attempt to fit the signal produced by three-way effects to closely matching pairwise effects, which are unlikely to be correct. While main effects are largely unaffected (Figure 4a) there is a significant increase in pairwise precision (Figure 4b). We also note that there is a trend toward poorer precision as we increase from main to pairwise and three-way effects. The number of candidates increases drastically at each step, and we do not currently remove combinations that produce identical columns from consideration. If there is a combination of effects that are present in exactly the same places as a true effect, these are indistinguishable and may be chosen instead. If this occurs, we consider it a false positive. Similarly, mostly overlapping three-way effects are far more common than mostly overlapping pairwise effects, and it is possible these may be chosen instead of a correct combination.

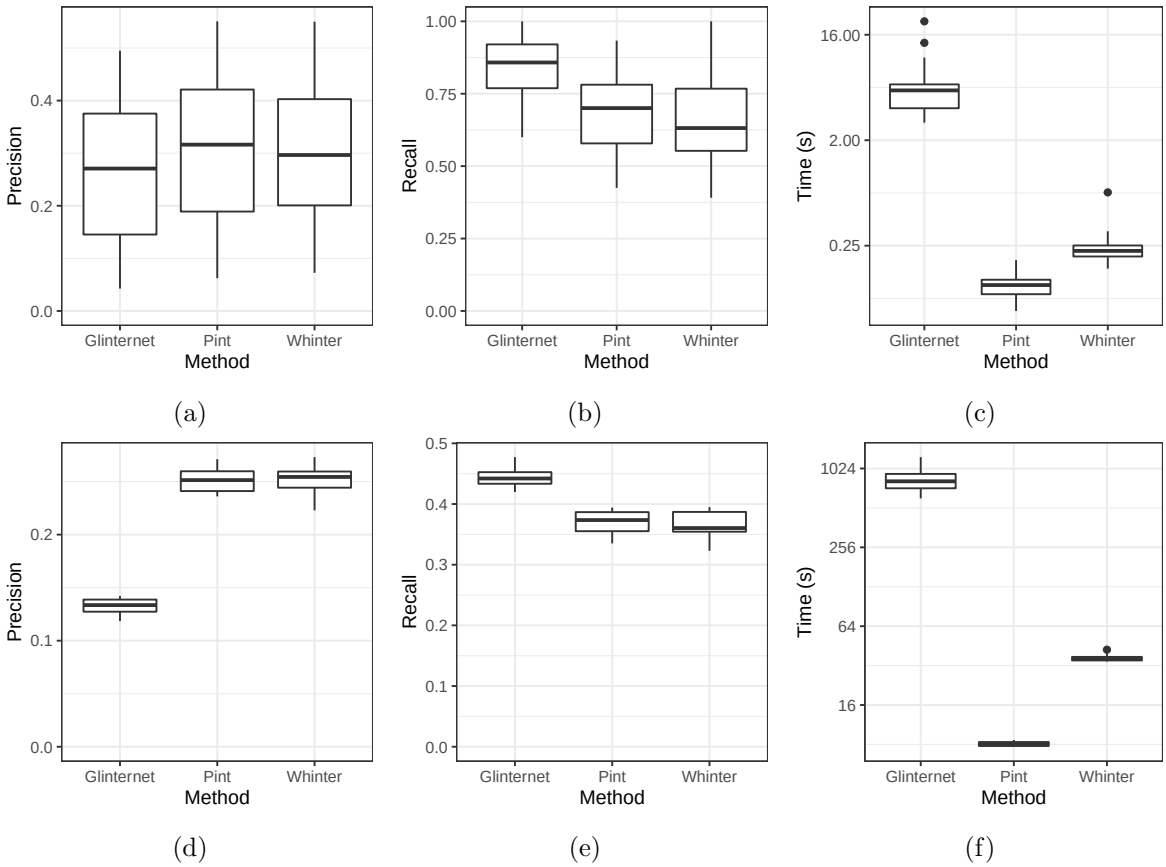


FIGURE 3. Searching for interactions with `glinternet` vs. WHInter vs. `Pint`. using data from [15]. Figures 3a to 3c $p = 100, n = 1,000$. Figures 3d to 3f $p = 4,000, n = 8,000$. Running time is measured using 32 threads on two Intel Xeon Gold 6244 CPUs, and shown on a log scale.

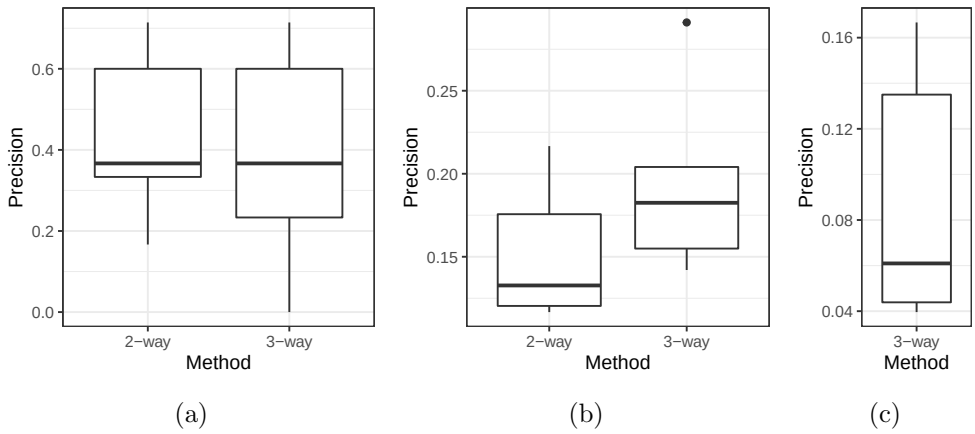


FIGURE 4. Precision of predicted large effects. (a) main, (b) pairwise, and (c) three-way. Data is fit with single, pairwise, and three-way interactions allowed (3-way), or only single and pairwise (2-way).

3.3. **InfectX siRNA Data.** We run our lasso model on the InfectX data (Section 2.7.1) allowing all pairwise and three-way interactions and halting at the end of the iteration once 100 non-zero effects are found. Only the combinations with non-zero predicted effects are then included in the matrix $\mathbf{Z}$. We then fit the fitness values $\mathbf{Y}$ to this matrix using least-squares

regression $\mathbf{Y} \sim \mathbf{Z}\beta$, using these unbiased estimates and p-values as our final result. The resulting fit has an adjusted R-squared of 0.13 and an AIC value of 18,184.82 We summarise the five most significant (according the p-value of the fit $\mathbf{Y} \sim \mathbf{Z}$) estimated effects in Table 1.

Table 1. InfectX most significant proposed effects.

Lasso Estimate	Least Squares Estimate	p-value	Gene 1	Gene 2	Gene 3
0.16930307	0.1420643	0.0001837726	PLCE1	—	—
0.31985354	0.2203255	0.0003231380	ZNF264	TRIM72	—
-0.09300632	-0.1599196	0.0005853407	TTC21A	—	—
0.14556737	0.1998505	0.0009281197	ANK1	KMT2D	ZHX3
-0.16530147	-0.1874086	0.0010551688	RNF213	SYN2	—

Among these effects, the three-way suppression of ANK1, KMT2D, and ZHX3 is particularly plausible. KMT2D is a known tumor suppressor, and mutations are common in lymphoma [61, 40]. ZHX3 is a transcriptional repressor, and in particular a failure of ZHX3 expression may be a cause of hepatocellular carcinoma [60]. Changes and failure to express KMT2D and ZHX3 respectively are associated with cancer development, and a significant increase in cell growth after suppressing both is consistent with these functions. ANK1 attaches integral membrane proteins to the cytoskeleton [57], and what role, if any, it plays is unclear.

The pairwise effect suppressing TRIM72 and ZNF264 could plausibly affect cell survival, as could suppressing PLCE1. PLCE1 is believed to play a role in cell survival and growth, and its suppression could have a significant effect on its own. It is unclear, however, how its suppression could have a positive effect on cell count [57]. TRIM72 plays a central role in cell membrane repair, and its suppression could easily affect fitness. ZNF264 may be involved in transcriptional regulation, and may have an interacting effect, although there are no known interactions between ZNF264 and TRIM72 [57].

The remaining two effects are TTC21A on its own, and RNF213 combined with SYN2. These genes are known to be involved in sperm function, vascular development, and neurotransmitter regulation respectively [35, 57]. We are not aware of any way in which RNF213 and SYN2 interact, or how either of these effects might affect cell growth or survival.

3.4. **Antibacterial Resistance.** As explained in Section 2.7.2, our antibacterial data is preprocessed to remove variants present in less than five cases, the remaining 174,334 columns are included in the model. We then fit the model $Y = X_2\beta + \epsilon$, where Y is the MIC log_2 phenotype (indicative of Ciprofloxacin resistance) and we include all pairwise interactions in X_2, stopping after 50 non-zero effects are found. These effects are included even with a large regularisation value λ, and are the most likely to be true positives [15]. Again creating a Z matrix with only the non-zero columns fitting $Y \sim Z$, we get a least-squares estimate with an Adjusted R-squared of 0.23.

The 50 non-zero effects involve 47 variants with 19 repeats and 16 pairwise effects (Appendix, Table 2). Of the pairwise effects, two pairs include the non-synonymous variant change that results in a Leu523Gln change in PA3054. PA3054 encodes a putative carboxypeptidase with the peptidase_M14 domain occurring between bases 24-634. Extracellular degradation of antimicrobials has been associated with increased production of M14 carboxypeptidases [53]. All other pairwise interactions identified involved synonymous variants. The most common of these was an A to G variant in PA3460, codon 537 Leu, found in 50% of the interactions. PA3460 encodes an acetyltransferase that is able to possibly modify Fluoroquinolones, reducing bacterial susceptibility to Ciprofloxacin [49]. The second most common synonymous variant was another A to G change in PA3709 encoding Ala 340 of a probable major facilitator superfamily (MFS) transporter protein. Overexpression of efflux pumps that include MFS transporters are associated with increased resistance to antibiotics [14, 39]. Despite none of the pairwise effects having known interactions, the majority variants identified are very poorly characterised so the possibility of a true combined effect of Ciprofloxacin resistance cannot be ruled out.

There were 34 variants that were characterised as contributing to Ciprofloxacin resistance. Of these, 16 were non-synonymous changes to proteins that are involved in fluoroquinolone modification, membrane transport or oxidative stress responses. The majority of non-synonymous variants occurred in oxidative stress response genes. An increase in reactive oxygen species (ROS) in response to Ciprofloxacin is well characterised in bacteria [23, 27, 1]. PA5401 encodes an electron transfer flavoprotein (EFT) domain-containing protein. The variant results in an Arg36Cys change in the β-subunit of an electron transfer protein whose gene is part of the dgc operon that is involved in choline metabolism and associated pathogenesis [18]. In eukaryotes, EFT is known to produce significant amounts of ROS in the presence of its partner enzyme medium-chain acyl-CoA dehydrogenase (MCAD) [50]. Therefore, non-synonymous variants in PA5401 could result in changes to pathogenesis or ROS amounts. Additional genes identified using our method include PA0117, pauD2, and gloA1 that are involved in glutathione production. Glutamine is a precursor of glutathione; glutamine and ascorbic acid have been found to provide substantial protection against Ciprofloxacin susceptibility in *Escherichia coli* [23].

Two genes with non-synonymous variants that were identified are involved in membrane integrity. The first is PA3173 with a His93Arg variation that encodes a short-chain dehydrogenase and acts on ubiG and ubiE involved in ubiquinone biosynthesis [28]. ROS accumulation affects membrane systems due to lipid peroxidation [17]. Ubiquinone is lipid-soluble and, therefore, is able to act as a mobile redox carrier within the cellular membrane [21]. Increased production of ubiquinone would reduce membrane damage caused by ROS. The second gene, MviN, with Leu316Met, is involved in peptidoglycan biosynthesis. Fosfomycin is frequently co-prescribed with Ciprofloxacin due to the synergistic activity of the two drugs [58]. However, increased peptidoglycan biosynthesis and cell wall recycling lead to antibiotic resistance [9]. Therefore the Leu316Met variant in MviN could be linked to increased resistance to combination therapy of Ciprofloxacin and Fosfomycin.

Overexpression of efflux pumps is a known contributor to increases in MICs for *P. aeruginosa*. Identified was a variant Lys329Gln in mexX that encodes a resistance-Nodulation-Cell Division (RND) multidrug efflux membrane fusion protein MexX precursor.

In summary, 13 of the 16 non-synonymous variants have possible contributions to Ciprofloxacin resistance.

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 and higher order 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, and no method we are aware of includes three-way interactions. In this paper we present a lasso regression based method for such large-scale inference of pairwise and three-way effects.

Our method effectively performs coordinate descent square-root lasso regression on a matrix containing all pairwise and three-way combinations of the input data. We expand upon the method used in [38], with a number of improvements. We update the working set in parallel, resulting in a significant speed improvement. We extend the method from two-way interactions to three-way, adding pruning of pairwise effects. We compress the active set with simple-8b, significantly reducing memory use and improving the running time with a large number of non-zero effects. Finally, we solve the square-root lasso instead of the lasso, giving us a well-defined stopping point.

We compared the accuracy and running time of our work to `glinternet`, the best of the methods we used previously [15] and WHInter, the fastest running method we are aware of [38]. Our simulations demonstrate comparable accuracy and recall to existing methods, running approximately three times faster than WHInter and 60 times faster than `glinternet` in the largest test. 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 15 billion possible pairwise interactions. In both cases our method finds a number of plausible interactions.

Despite this success, our method and its implementation in `Pint` have the following limitations. While our method is effective on genome scale data when using only pairwise interactions, running time limits the use of three-way interactions to smaller sets, or only the strongest interactions (running to completion was only possible with $p \leq 5,000$ in our testing). Furthermore, the sub-problem given the working set is not solved in parallel. While it is possible to do so, it is actually harmful to performance unless the working set is very large (i.e. many non-zero effects are included).

Note that while we consider only pairwise interactions Section 3.4, it is possible to include three-way effects if we remove columns with less than 30 entries instead. This reduces the input from $174,334$ to $75,599$ columns, and the first 50 interactions can then be found in approx. 80 hours.

The additive interaction model is also an oversimplification of biology. It remains unclear to what extent genetic effects be treated as additive, and ignoring interactions among of more than three items could well be leaving out the most important effects. In this case we may end up spuriously associating phenotype changes with effects that just happen to be present, rather than the true, more complicated, interactions (as demonstrated in Section 3.2). Finally, we cannot distinguish interactions that are present in exactly the same rows of the input matrix. In the siRNA case, if two distinct pairs of genes are simultaneously suppressed by the same siRNAs in all cases, whichever is considered first will likely trump the other.

There are hence a number of opportunities to expand upon this work. Grouping indistinguishable effects would not only reduce the running time, but make the groups of indistinguishable interactions clear. Our implementation currently requires the input matrix $\mathbf{X}$ to be binary, but this limitation could be removed by storing a list of all entry values, and including these in matrix operations. Finally, we chose the square-root lasso penalty partially because it has simpler to compute p-values than the lasso [55]. Implementing this would give unbiased p-values based directly on the lasso results, without requiring a second least squares fit. This is particularly important since the least squares p-values do not account for the column selection process, and are likely to be biased [32].

More generally, we could significantly increase the scale of interaction inference methods by reducing the search space. A more targeted approach estimating distance in 3D space using Hi-C [7] for example, would drastically reduce the time and space requirements, allowing higher order interactions to be considered. A weak hierarchy assumption would allow us to ignore all interactions where neither of the main effects are (yet) non-zero. Strong hierarchy would further simplify the problem, although these are not reasonable assumptions for all applications. Finally, the interactions proposed in Section 3.3 may be worth further investigation.

Our method is implemented in C++, and an R package is provided at github.com/bioDS/pint. Code to reproduce the simulations and benchmarking from Section 3 is provided at github.com/bioDS/lasso_data_processing.

References

[1] Marwa N. Ahmed et al. "Evolution of Antibiotic Resistance in Biofilm and Planktonic Pseudomonas Aeruginosa Populations Exposed to Subinhibitory Levels of Ciprofloxacin". *Antimicrobial Agents and Chemotherapy* 62.8 (2018), e00320–18. DOI: 10.1128/AAC.00320-18.

[2] Ferhat Alkan et al. "RIsearch2: Suffix Array-Based Large-Scale Prediction of RNA–RNA Interactions and siRNA off-Targets". *Nucleic Acids Research* 45.8 (May 5, 2017), e60–e60. DOI: 10.1093/nar/gkw1325.

[3] Vo Ngoc Anh and Alistair Moffat. "Inverted Index Compression Using Word-Aligned Binary Codes". *Information Retrieval* 8.1 (Jan. 1, 2005), pp. 151–166. DOI: 10.1023/B:INRT.0000048490.99518.5c.

[4] Alan Ashworth, Christopher J. Lord, and Jorge S. Reis-Filho. "Genetic Interactions in Cancer Progression and Treatment". *Cell* 145.1 (Apr. 1, 2011), pp. 30–38. DOI: 10.1016/j.cell.2011.03.020.

[5] N Beerenwinkel, L Pachter, and B Sturmfels. "Epistasis and Shapes of Fitness Landscapes". *Statistica Sinica* (2007).

[6] A. Belloni, V. Chernozhukov, and L. Wang. "Square-Root Lasso: Pivotal Recovery of Sparse Signals via Conic Programming". *Biometrika* 98.4 (Dec. 1, 2011), pp. 791–806. DOI: 10.1093/biomet/asr043.

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

[8] Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information and Citation Key: GeneInternet. *Gene [Internet]*. URL: https://www.ncbi.nlm.nih.gov/gene/.

[9] Marina Borisova, Jonathan Gisin, and Christoph Mayer. "Blocking Peptidoglycan Recycling in Pseudomonas Aeruginosa Attenuates Intrinsic Resistance to Fosfomycin". *Microbial Drug Resistance* 20.3 (June 1, 2014), pp. 231–237. DOI: 10.1089/mdr.2014.0036.

[10] João Botelho, Filipa Grosso, and Luísa Peixe. "Antibiotic Resistance in Pseudomonas Aeruginosa – Mechanisms, Epidemiology and Evolution". *Drug Resistance Updates* 44 (May 1, 2019), p. 100640. DOI: 10.1016/j.drup.2019.07.002.

[11] 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.

[12] Michael Costanzo et al. "The Genetic Landscape of a Cell." *Science* (2010). pmid: 20093466.

[13] Kristina Crona et al. "Inferring Genetic Interactions from Comparative Fitness Data". *Elife* 6 (Dec. 2017).

[14] Edda De Rossi et al. "The Multidrug Transporters Belonging to Major Facilitator Superfamily (MFS) in Mycobacterium Tuberculosis". *Molecular Medicine* 8.11 (11 Nov. 2002), pp. 714–724. DOI: 10.1007/BF03402035.

[15] Kieran Elmes et al. "Learning Epistatic Gene Interactions from Perturbation Screens". *PLOS ONE* 16.7 (July 13, 2021), e0254491. DOI: 10.1371/journal.pone.0254491.

[16] D. S. (Douglas Scott) Falconer. *Introduction to Quantitative Genetics*. 4th ed. Harlow, Essex, England: Longman, 1996. ISBN: 0-582-24302-5.

[17] Edward E. Farmer and Martin J. Mueller. "ROS-Mediated Lipid Peroxidation and RES-Activated Signaling". *Annual Review of Plant Biology* 64.1 (2013), pp. 429–450. DOI: 10.1146/annurev-arplant-050312-120132. pmid: 23451784.

[18] Liam F. Fitzsimmons et al. "Small-Molecule Inhibition of Choline Catabolism in Pseudomonas Aeruginosa and Other Aerobic Choline-Catabolizing Bacteria". *Applied and Environmental Microbiology* 77.13 (July 1, 2011), pp. 4383–4389. DOI: 10.1128/AEM.00504-11.

[19] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". *Journal of Statistical Software* 33.1 (2010). DOI: 10.18637/jss.v033.i01. pmid: 20808728.

[20] Wenjiang J. Fu. "Penalized Regressions: The Bridge versus the Lasso". *Journal of Computational and Graphical Statistics* 7.3 (Sept. 1998), pp. 397–416. DOI: 10.1080/10618600.1998.10474784.

[21] Naoko Fujimoto, Tomoyuki Kosaka, and Mamoru Yam. "Menaquinone as Well as Ubiquinone as a Crucial Component in the Escherichia Coli Respiratory Chain". *Chemical Biology*. Ed. by Deniz Ekinci. InTech, Feb. 17, 2012. ISBN: 978-953-51-0049-2. DOI: 10.5772/35809.

[22] Robert Gaynes, Jonathan R. Edwards, and National Nosocomial Infections Surveillance System. "Overview of Nosocomial Infections Caused by Gram-Negative Bacilli". *Clinical Infectious Diseases* 41.6 (Sept. 15, 2005), pp. 848–854. DOI: 10.1086/432803.

[23] M. Goswami, S. H. Mangoli, and N. Jawali. "Involvement of Reactive Oxygen Species in the Action of Ciprofloxacin against Escherichia Coli". *Antimicrobial Agents and Chemotherapy* 50.3 (Mar. 1, 2006), pp. 949–954. DOI: 10.1128/AAC.50.3.949-954.2006.

[24] Alison L Gould et al. "Microbiome Interactions Shape Host Fitness". *Proceedings of the National Academy of Sciences of the United States of America* 115.51 (Dec. 2018), E11951–E11960.

[25] *GRCh38.P13 - Genome - Assembly - NCBI*. URL: https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.39.

[26] *IPCD International Pseudomonas Consortium Database*. URL: https://ipcd.ibis.ulaval.ca/.

[27] Peter Ø. Jensen et al. "Formation of Hydroxyl Radicals Contributes to the Bactericidal Activity of Ciprofloxacin against Pseudomonas Aeruginosa Biofilms". *Pathogens and Disease* 70.3 (Apr. 1, 2014), pp. 440–443. DOI: 10.1111/2049-632X.12120.

[28] Hai-Xia Jiang et al. "Coenzyme Q Biosynthesis in the Biopesticide Shenqinmycin-producing Pseudomonas Aeruginosa Strain M18". *Journal of Industrial Microbiology and Biotechnology* 46.7 (July 1, 2019), pp. 1025–1038. DOI: 10.1007/s10295-019-02179-1.

[29] Timothy J. Kidd et al. "Pseudomonas Aeruginosa Exhibits Frequent Recombination, but Only a Limited Association between Genotype and Ecological Setting". *PLoS ONE* 7.9 (Sept. 6, 2012). Ed. by Sam Paul Brown, e44199. DOI: 10.1371/journal.pone.0044199.

[30] Ben Langmead and Steven L. Salzberg. "Fast Gapped-Read Alignment with Bowtie 2". *Nature Methods* 9.4 (Apr. 2012), pp. 357–359. DOI: 10.1038/nmeth.1923.

[31] Jeff Larkin. "OpenMP on GPUs, First Experiences and Best Practices" (Nvidia GTC 2018).

[32] Jason D. Lee et al. "Exact Post-Selection Inference, with Application to the Lasso". *The Annals of Statistics* 44.3 (June 2016), pp. 907–927. DOI: 10.1214/15-AOS1371.

[33] D. Lemire and L. Boytsov. "Decoding Billions of Integers per Second through Vectorization". *Software: Practice and Experience* 45.1 (2015), pp. 1–29. DOI: 10.1002/spe.2203.

[34] Caitlin Lienkaemper et al. "The Geometry of Partial Fitness Orders and an Efficient Method for Detecting Genetic Interactions". *Journal of Mathematical Biology* 77.4 (May 2018), pp. 951–970.

[35] Wangjie Liu et al. "Bi-Allelic Mutations in TTC21A Induce Asthenoteratospermia in Humans and Mice". *American Journal of Human Genetics* 104.4 (Apr. 4, 2019), pp. 738–748. DOI: 10.1016/j.ajhg.2019.02.020. pmid: 30929735.

[36] 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". *Microbes and Infection* 2.9 (July 1, 2000), pp. 1051–1060. DOI: 10.1016/S1286-4579(00)01259-4.

[37] Antonio Mallia, Michał Siedlaczek, and Torsten Suel. "An Experimental Study of Index Compression and DAAT Query Processing Methods". *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.

[38] Marine Le Morvan and Jean-Philippe Vert. "WHInter: A Working Set Algorithm for High-Dimensional Sparse Second Order Interaction Models." (), p. 10.

[39] Ankita Nag and Sarika Mehra. "A Major Facilitator Superfamily (MFS) Efflux Pump, SCO4121, from Streptomyces Coelicolor with Roles in Multidrug Resistance and Oxidative Stress Tolerance and Its Regulation by a MarR Regulator". *Applied and Environmental Microbiology* 87.7 (), e02238–20. DOI: 10.1128/AEM.02238-20.

[40] Ana Ortega-Molina et al. "The Histone Lysine Methyltransferase KMT2D Sustains a Gene Expression Program That Represses B Cell Lymphoma Development". *Nature Medicine* 21.10 (10 Oct. 2015), pp. 1199–1208. DOI: 10.1038/nm.3943.

[41] Jakub Otwinowski and Ilya Nemenman. "Genotype to Phenotype Mapping and the Fitness Landscape of the E. Coli Lac Promoter". *PLoS ONE* 8.5 (May 1, 2013). Ed. by Andrew J. Yates, e61570. DOI: 10.1371/journal.pone.0061570.

[42] Preeti Pachori, Ragini Gothalwal, and Puneet Gandhi. "Emergence of Antibiotic Resistance Pseudomonas Aeruginosa in Intensive Care Unit; a Critical Review". *Genes & Diseases* 6.2 (June 1, 2019), pp. 109–119. DOI: 10.1016/j.gendis.2019.04.001.

[43] Zheng Pang et al. "Antibiotic Resistance in Pseudomonas Aeruginosa: Mechanisms and Alternative Therapeutic Strategies". *Biotechnology Advances* 37.1 (Jan. 1, 2019), pp. 177–192. DOI: 10.1016/j.biotechadv.2018.11.013.

[44] powturbo. *Powturbo/TurboPFor-Integer-Compression*. July 9, 2020.

[45] Pauli Rämö et al. "Simultaneous Analysis of Large-Scale RNAi Screens for Pathogen Entry". *BMC Genomics* 15.1 (Dec. 22, 2014), p. 1162. DOI: 10.1186/1471-2164-15-1162.

[46] 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". *Journal of Medical Microbiology* 68.11 (Nov. 1, 2019), pp. 1591–1595. DOI: 10.1099/jmm.0.001085.

[47] Attika Rehman, Wayne M. Patrick, and Iain L. Lamont. "Mechanisms of Ciprofloxacin Resistance in Pseudomonas Aeruginosa: New Approaches to an Old Problem". *Journal of Medical Microbiology,* 68.1 (2019), pp. 1–10. DOI: 10.1099/jmm.0.000873.

[48] Tracey Remmington, Nikki Jahnke, and Christian Harkensee. "Oral Anti-Pseudomonal Antibiotics for Cystic Fibrosis". *Cochrane Database of Systematic Reviews* (July 14, 2016). Ed. by Cochrane Cystic Fibrosis and Genetic Disorders Group. DOI: 10.1002/14651858.CD005405.pub4.

[49] Ari Robicsek et al. "Fluoroquinolone-Modifying Enzyme: A New Adaptation of a Common Aminoglycoside Acetyltransferase". *Nature Medicine* 12.1 (1 Jan. 2006), pp. 83–88. DOI: 10.1038/nm1347.

[50] João V. Rodrigues and Cláudio M. Gomes. "Mechanism of Superoxide and Hydrogen Peroxide Generation by Human Electron-Transfer Flavoprotein and Pathological Variants". *Free Radical Biology and Medicine* 53.1 (July 1, 2012), pp. 12–19. DOI: 10.1016/j.freeradbiomed.2012.04.016.

[51] Benjamin Schlegel, Rainer Gemulla, and Wolfgang Lehner. "Fast Integer Compression Using SIMD Instructions". *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.

[52] Fabian Schmich et al. "gespeR: A Statistical Model for Deconvoluting off-Target-Confounded RNA Interference Screens". *Genome Biology* 16.1 (Oct. 7, 2015), p. 220. DOI: 10.1186/s13059-015-0783-1.

[53] Guomeng Sha et al. "Dynamics and Removal Mechanisms of Antibiotic and Antibiotic Resistance Genes during the Fermentation Process of Spectinomycin Mycelial Dregs: An Integrated Meta-Omics Study". *Journal of Hazardous Materials* 421 (Jan. 5, 2022), p. 126822. DOI: 10.1016/j.jhazmat.2021.126822.

[54] *STRING: Functional Protein Association Networks*. URL: https://string-db.org/cgi/about.pl.

[55] Xiaoying Tian, Joshua R Loftus, and Jonathan E Taylor. "Selective Inference with Unknown Variance via the Square-Root Lasso". *Biometrika* 105.4 (Dec. 1, 2018), pp. 755–768. DOI: 10.1093/biomet/asy045.

[56] Andrew Trotman and Jimmy Lin. "In Vacuo and In Situ Evaluation of SIMD Codecs". *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.

[57] *UniProt: The Universal Protein Knowledgebase in 2021 — Nucleic Acids Research — Oxford Academic*. URL: https://academic.oup.com/nar/article/49/D1/D480/6006196.

[58] Lei Wang et al. "Synergistic Activity of Fosfomycin, Ciprofloxacin, and Gentamicin Against Escherichia Coli and Pseudomonas Aeruginosa Biofilms". *Frontiers in Microbiology* 10 (2019), p. 2522. DOI: 10.3389/fmicb.2019.02522.

[59] Tong Tong Wu and Kenneth Lange. "Coordinate Descent Algorithms for Lasso Penalized Regression". *The Annals of Applied Statistics* 2.1 (Mar. 2008), pp. 224–244. DOI: 10.1214/07-AOAS147. arXiv: 0803.3876.

[60] Kazuya Yamada et al. "ZHX2 and ZHX3 Repress Cancer Markers in Normal Hepatocytes". *Frontiers in bioscience (Landmark edition)* 14 (Jan. 1, 2009), pp. 3724–3732. DOI: 10.2741/3483. pmid: 19273305.

[61] Jiyuan Zhang et al. "Disruption of KMT2D Perturbs Germinal Center B Cell Development and Promotes Lymphomagenesis". *Nature Medicine* 21.10 (10 Oct. 2015), pp. 1190–1198. DOI: 10.1038/nm.3940.

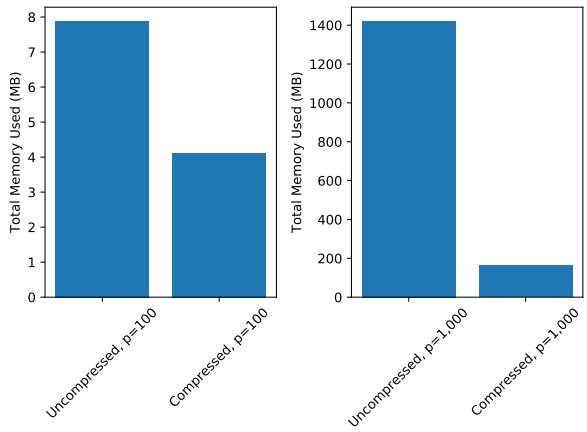


FIGURE 5. Compression effect on memory use. Note that this is the total peak memory use of the program, not solely the memory used by the matrix $\mathbf{X}_2$. In both cases $n = 10 \cdot p$.

APPENDIX A. COMPRESSION

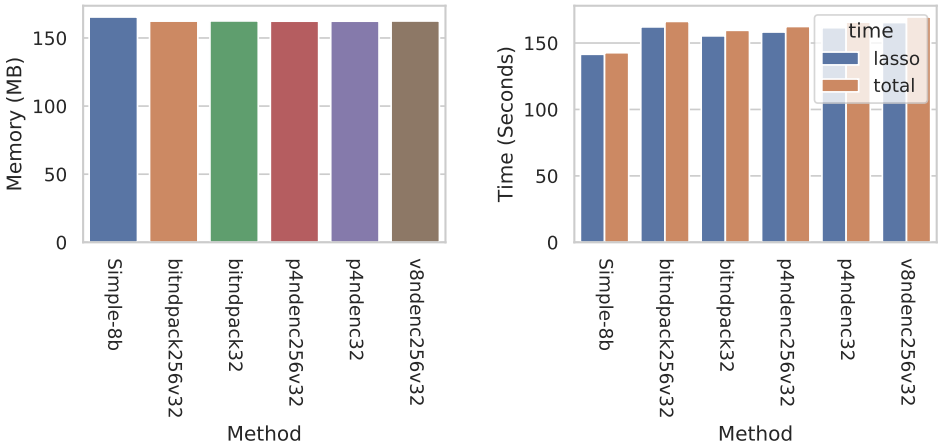
We can considerably reduce the size of the active set by compression the columns. Since we have a sequence of increasing integers we can store only the offset from the previous entry, keeping the entries small. The resulting sequence of (mostly) small numbers can then be efficiently stored using integer compression methods. We describe the compression method we use in Appendix A.1 and compare it to other methods in Appendix A.2.

A.1. **Simple-8b.** Simple-8b is a non-SIMD compression scheme, with performance comparable to other state of the art methods [51, 37, 56]. While SIMD-based compression schemes can often offer significantly improved compression and decompression speed [33] [51], their implementation is architecture dependant. Simple-8b only requires a CPU be able to efficiently handle 64-bit arithmetic, and does not significantly underperform compared to state-of-the-art SIMD techniques in our testing (Appendix A.2).

Simple-8b is a 64-bit variation of the Simple-9 encoding scheme [3], and stores a sequence of integers in a single 64-bit word. The number of integers stored depends on the size of the largest one, and is indicated by a four bit 'selector'. The remaining 60 bits are divided into integers of size $1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 15, 20, 30$, or 60, with between 240 (only possible if all values are zero) and one integer stored. As seen in Figure 5, this considerably reduces the size of $\mathbf{X}_2$ in our test data (two sets from [15], one with $p = 100$, $n = 1,000$, another with $p = 1,000$, $n = 10,000$). In the larger $p = 1,000$ set, total memory use is reduced by over 85% compared to storing integers directly. It is worth noting that this compression works well even for non-sparse sections of the matrix, since the offsets are extremely small. In an extreme case, we can store up to 240 sequential 1's in a single 64-bit word.

A.2. **Comparing Methods.** While Simple-8b allows our implementation to be used on any 64-bit CPU, we could also take advantage of SIMD-based methods where such CPU instructions are available. To determine whether this is a worthwhile improvement, we compare our Simple-8b implementation to a number of state of the art alternatives.

Recent work suggests TurboPFor [44] has a particularly high compression ratio [56]. We therefore compare the best performing methods from TurboPFor against our implementation of Simple-8b (Figure 6). The tests are performed using an eight-core (16 SMT threads) Intel Xeon Gold 6244 CPU. To compare these methods, we perform 50 regression iterations on a test data set of $p = 1,000$ genes and $n = 10,000$ siRNAs. We examine the total time taken for the process, as well as the total memory used and time for the regression function alone (excluding calculating and compressing the interaction matrix).



(a) (b)

FIGURE 6. Comparison of Compression Methods. (a) Total memory used, compressing the sparse $\mathbf{X}_2$ matrix with each method. (b) Total time taken and time taken (including compressing $\mathbf{X}_2$) and time taken for lasso regression alone, using each method.

We see that both the time to produce the compressed matrix (seen in Figure 6 as the difference between total time and lasso-only time), and the running time are comparable for all TurboPFor methods.[2] While every TurboPFor method we tested improved the compression ratio compared to Simple-8b (Figure 6a), we consistently found that the running time was longer (Figure 6b). It is possible that this is a result of the way the columns are being read in each method. Using TurboPFor, we compress and decompress entire columns at a time. With our Simple-8b implementation, we process each 64-bit word separately. This allows us to use the column as it is being decompressed. Avoiding re-reading the column after decompression also allows the entries to be evicted from the cache earlier.

While it is also possible to process compressed words as they are read using the tested TurboPFor methods, there does not appear to be a significant difference in compression that would justify doing so.

[2]The compression time is not comparable for all methods. Our Simple-8b implementation compresses columns in parallel, whereas TurboPFor does not. Columns are decompressed in parallel in both cases

Appendix B. Antibiotic Results

Table 2. Predicted top 50 SNV effects.

Lasso Estimate	Least Squares Estimate	pval	SNV 1	SNV 2
-0.024	-1.76	0.01	813809 sub. T C	—
-0.013	-5.02	0.02	3789873 sub. T C	—
-0.039	-1.68	0.02	4617770 sub. C T	—
-0.007	9.71	0.03	5111434 sub. A G	—
-0.006	-2.83	0.03	4153211 sub. A G	5719820 sub. A G
0.000	-1.15	0.05	6081497 sub. T C	—
-0.007	-2.25	0.06	4271404 sub. T C	—
-0.023	-2.25	0.06	63407 sub. T C	—
-0.063	-2.25	0.06	184347 sub. T C	—
-0.001	-12.62	0.07	137771 sub. A G	—
-0.008	-2.76	0.10	3654726 sub. T C	3869317 sub. A G
-0.019	-1.03	0.11	1886446 sub. T C	—
-0.045	13.18	0.11	442184 sub. C G	—
-0.001	-2.10	0.12	986363 sub. A G	—
-0.003	-1.84	0.13	2996864 sub. C T	3869317 sub. A G
-0.030	-1.60	0.16	1810265 sub. A C	—
-0.012	0.76	0.20	2033361 sub. A C	—
-0.009	-7.03	0.22	3318136 sub. T C	—
-0.038	8.19	0.23	176977 sub. A G	—
-0.002	2.44	0.24	172758 sub. T C	—
-0.009	2.00	0.24	4003269 sub. T C	—
-0.034	8.48	0.24	4153211 sub. A G	—
-0.008	-6.99	0.28	3419855 sub. T A	4153211 sub. A G
-0.031	-1.25	0.30	4847782 sub. A C	—
-0.005	-1.25	0.30	1383522 sub. A G	—
-0.005	-1.25	0.30	176977 sub. A G	5133603 sub. A G
-0.004	2.36	0.30	3419855 sub. T A	4920968 sub. C A
-0.040	2.44	0.31	3944454 sub. T A	—
-0.004	-0.84	0.40	3195104 sub. T C	—
-0.028	-0.42	0.40	2211528 sub. T G	—
-0.013	2.14	0.51	313283 sub. A C	—
0.000	0.90	0.51	3601029 sub. T C	3869317 sub. A G
-0.005	4.38	0.53	3419855 sub. T A	—
-0.010	2.64	0.55	5254860 sub. T C	—
-0.016	-0.31	0.69	2093685 sub. A T	—
-0.007	1.02	0.73	3869317 sub. A G	4687756 sub. A G
-0.044	-0.59	0.78	4453098 sub. A G	—
-0.005	0.24	0.80	3563823 sub. A G	—
-0.039	1.21	0.81	4125422 sub. T C	—
-0.002	0.57	0.88	3869317 sub. A G	4966957 sub. C T
-0.103	0.21	0.95	3869317 sub. A G	—
-0.005	—	1.00	956627 sub. A G	—
-0.001	—	1.00	5856921 sub. A G	—
-0.002	—	1.00	3869317 sub. A G	4617770 sub. C T
-0.014	—	1.00	3869317 sub. A G	4847782 sub. A C
-0.008	—	1.00	4153211 sub. A G	4453098 sub. A G
-0.046	—	1.00	3869317 sub. A G	4453098 sub. A G
-0.006	—	1.00	4153211 sub. A G	5990479 sub. C G
-0.009	—	1.00	3195104 sub. T C	4153211 sub. A G
-0.046	—	1.00	986363 sub. A G	3869317 sub. A G