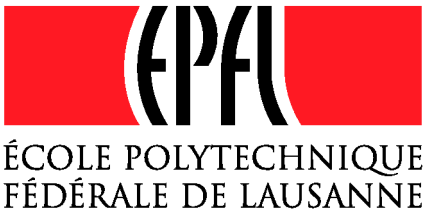

Master's Thesis

Using a random walker on gene expression and protein-protein interaction networks to prioritize candidate genes

Author:
Jonas Helfer

Supervisors:
Prof Bernard Moret
Prof Rui Jiang

Beijing, August 2011

Abstract

Identification of genes underlying human diseases is an important step in understanding and treating genetic disorders. Based on the assumption that related diseases are caused by related genes, several methods for candidate gene prioritization have been proposed in the past to refine lists of suspect genes obtained by linkage analysis or other methods. The large increase in publicly available -omics data has made it possible to implement prioritization methods that combine information from multiple data sources to make better rankings. In this work, we present a new method for prioritization of candidate disease genes based on gene expression data, that ranks 12851 genes for 5080 phenotypes. The performance is comparable to previous methods which used hand-curated protein-protein data on smaller test sets. We also propose a method for combining multiple gene networks into a single one with which we ranked up to 14612 genes for 5080 phenotypes, more than any previous method. Our evaluation shows, that the performance of the fused network is superior to that of its separate component networks. In an effort to assure reproducibility of results, all the code written for this research was made public and is freely available to anyone wishing to use or extend it in any way.

Keywords: Disease gene, gene expression, network, protein-protein interaction, phenotype similarity, random walker

Contents

1 Introduction 1
1.1 Between biology and computer science . . . 1
1.2 Finding disease genes . . . 1

2 Materials and Methods 3
2.1 Data sources . . . 3
2.1.1 Phenotype data . . . 3
2.1.2 Gene expression data . . . 3
2.1.3 Protein-protein interaction data . . . 3
2.1.4 Phenotype-gene associations . . . 3
2.1.5 Artificial linkage interval . . . 4
2.2 Constructing phenotype and gene networks . . . 4
2.3 Random walker with restart . . . 5
2.4 Random walk on the heterogeneous network . . . 5
2.5 Fusing gene networks . . . 7
2.6 Validation methods . . . 7
2.6.1 Leave-one-out cross-validation . . . 7
2.6.2 Ab-initio cross-validation . . . 7
2.7 Performance measures . . . 8
2.7.1 ROC . . . 8
2.7.2 Enrichment . . . 8

3 Implementation 9
3.1 Platform . . . 9
3.2 Packages . . . 10

4 Results 11
4.1 Network characteristics . . . 11
4.2 Performance of GE networks . . . 14
4.3 Fused network . . . 18
4.4 Comparison of fused network with other methods . . . 24

5 Comparison and reproducibility of results 25

6 Conclusion 27

7 Acknowledgements 29

References 31

8 Appendix A
8.1 Input files . . . A
8.2 Tables of matrix comparisons . . . A

List of Figures

1 Distribution of the correlation coefficients for the expression of 12851 genes in 158 tissues 12

2 Total number of edges in GE network for different thresholds 12

3 Box plot of the number of connections per node for gene expression networks with correlation threshold 0.5,0.6,0.7,0.8 and 0.9 and the PPI network 13

4 Histogram of the number of connections for the PPI network and the gene expression network with correlation threshold 0.5. The PPI network has 9607 nodes and the gene expression network has 12851 nodes. 13

5 Comparison of the number of perfect rankings between different phenotype and gene networks 15

6 Comparison of the number of null-predictions between different phenotype and gene networks 15

7 AUC scores for different combinations of phenotype and gene networks) 16

8 Artificial linkage interval ROC curves for RWRH ranking on the phenotype network with threshold 0.6 and the gene network with threshold 0.9, once ignoring null-predictions (green) and once assigning random rank to null-predictions (blue). 17

9 Artificial linkage interval ROC curves for RWRH ranking using gene expression networks with different thresholds and KNN values in combination with a symmetrical 9-KNN phenotype network. 17

10 Comparison of ROC curves for RWRH leave-one out cross-validation rankings with PPI network, GE network with threshold 0.5 and the two networks fused together (α=0.3). There were 1717,1630 and 1428 gene-phenotype connections for the fused network, the GE network and the PPI network respectively. 19

11 Scatterplot of rank of fused network plotted against rank of GE network only for leave-one-out cross validation (1630 connections) on an artificial linkage interval of 100 genes. Values on the diagonal mean equal rank was assigned. The blue line shows the best linear fit (with minimal square error), indicating that the performance of RWRH on the fused network is superior to that of RWRH on the gene expression network alone. 20

12 Scatterplot of rank of fused network plotted against rank of PPI network only for leave-one-out cross validation (1428 connections) on artificial linkage interval of 100 genes. Values on the diagonal mean equal rank was assigned. The blue line shows the best linear fit (minimal square error), indicating that the performance of RWRH on the fused network is superior to that of RWRH on the protein-protein interaction network alone. 21

13 AUC scores for the three networks for leave-one-out cross validation on artificial linkage interval, whole-genome leave-one-out cross-validation and *ab-initio* cross-validation. The number of validation cases were 1717,1428 and 1630 for the three different networks. . . . 22

14 ROC curves for the three networks for whole-genome leave-one-out cross-validation. There validation set included 1717,1630 and 1428 gene-phenotype connections for the fused network, the GE network and the PPI network respectively. 23

List of Tables

1 AUC values for leave-one out cross-validation on an artificial linkage interval of 100 genes . . . . . B
2 null-prediction counts for leave-one out cross-validation on an artificial linkage interval of 100 genes C
3 Perfect ranking counts for leave-one out cross-validation on an artificial linkage interval of 100 genes . . . . . D
4 Average enrichment scores for leave-one out cross-validation on an artificial linkage interval of 100 genes . . . . . E
5 AUC scores for leave-one out cross-validation on the whole genome F
6 null-prediction counts for leave-one out cross-validation on the whole genome . . . . . G
7 Perfect ranking counts for leave-one out cross-validation on the whole genome . . . . . H
8 Average enrichment scores for leave-one out cross-validation on the whole genome I
9 AUC scores for parameter tests on artificial linkage interval . . . . . J
10 AUC scores for parameter tests on the whole genome . . . . . J

1 Introduction

This section provides a background to situate the core of the thesis and gives references to more detailed works on the concepts introduced.

1.1 Between biology and computer science

For centuries biology used to be a purely descriptive science, mainly occupying itself with the task of describing the variety of organisms on earth, so vast in numbers that it has occupied scores of zoologists and botanists for centuries. While this branch of biology is still important today, advances in physics and chemistry have made it possible to study life on a molecular level, offering insights and giving explanations for things that the scientists of previous centuries had to accept as given. With new techniques such as electron microscopy, radiocristallography, DNA sequencing and many more, the limits of the explainable are pushed to ever smaller scales. While chemistry and sometimes physics are employed to find out 'how', the theory of evolution provides a means to answer the question 'why?'

However, as the boundary of the visible and measurable was pushed to ever smaller scales, it has proven very difficult to match the pace with our understanding. Until shortly before the end of the 20th century, discoveries were usually made by those, who had the best data. With the introduction of so-called high-throughput methods, this changed radically. Now, there is a vast amount of data available, from which it is increasingly difficult to extract all the possible knowledge. Inevitably, this shift gave rise to a new kind of scientist, who - rather than staring through a microscope or counting flies in tubes - sits behind a screen and wields the growing power of computers and algorithms to extract as much knowledge as possible from the mountains of data.

1.2 Finding disease genes

The great amount of -omics data that is now publicly available in different online-databases can be used to answer many different questions. One such question concerns itself with the hereditary causes of human diseases. For medicine and pharmacology, it is of crucial interest to understand the mechanisms underlying a disorder in order to treat it most effectively. In the case of disorders with a hereditary component, finding the causative genes is a first important step. Currently, over 1700 inherited disorders with unknown genetic origins are listed in the Online Mendelian Inheritance in Man (OMIM) database [1]. An additional 1900 are listed with suspected (but not proven) genetic origins. For comparison, the number of disorders with known origin in OMIM is currently around 4400.

Linkage analysis is usually the first step towards tying a gene to an inherited disorder. It allows researchers to identify statistically significant co-inheritance of genetic markers and the disorder in question, pointing to a rough chromosomal location of the genetic factors involved. Usually, current linkage studies are able to identify an interval of 0.5 to 10 cM, containing up to 300 genes [2]. While genome wide association studies (GWAS) can in many cases pinpoint the gene involved, they are costly, time consuming and require careful selection of genes and subjects [3, 4]. In recent years several different computational methods allowing the refinement of candidate gene lists have therefore been proposed.

Early computational methods for candidate gene prioritization typically ranked genes based on their similarity to known causative genes. Peretz-Iratxeta et al [5] were among the first to propose a data-mining method based on functional annotation data. Other methods used sequence based features [6, 7] or protein-protein interaction data [8, 9]. These methods offer a significant improvement over random rankings, but they also have a number of limitations. First of all, it is difficult to define a clear boundary between two diseases. A disease usually involves several biological pathways and expresses itself in more than one phenotype. Leber's congenital amaurosis (LCA) for instance clinically appears to be one single disease, but is in fact caused by a group of very different defects on a molecular level [10]. Such fuzzy boundaries make it impossible to derive direct and non-ambiguous gene-disease associations. Another difficulty with these methods is the noise and incompleteness of genomic and proteomic data.

A number of algorithms have been proposed that try to sidestep these difficulties by relying on gene-phenotype relationships and using multiple data sources. The first such method, ENDEAVOUR, is based on order statistics to fuse ranks obtained on as many as 10 data sets into one [11]. Other algorithms predicting gene-phenotype relationships make extensive use of available information about biological networks and calculate a distance measure of one sort or another to generate a ranking. CIPHER combines phenotype and genotype networks and prioritizes genes based on a concordance score of their gene-phenotype profile. Köhler et al proposed an algorithm based on a random walker model on a protein-protein interaction network [8]. Li & Patra later extended this work to include a phenotype network [12]. Other approaches recently tried on disease gene prioritization include algorithms based on network propagation, electrical flow and Bayesian regression models [13, 14, 15]. All these methods implicitly use the fact that human diseases are modular in nature, i.e. that related diseases tend to be caused by related genes [16].

Unfortunately, many of methods that have been proposed prioritize only a small number of manually curated genes. ENDEAVOUR for example, which used as many as 10 data sources, ranks only 672 genes for a very specific test set. Other methods have greatly improved on this, but so for none of them produce a ranking for all the genes of the human genome.

2 Materials and Methods

2.1 Data sources

Data sources used in this work include the mimMiner phenotype similarity data[17] for 5080 phenotypes, a list of 9607 binary protein-protein interactions from the Human Protein Reference Database (HPRD) [18], gene expression data for 33689 markers in 158 tissues from microarray experiments by Su et al [19], as well as a list of 7106 connections between 2463 disease phenotypes and 4528 associated genes from the Online Mendelian Inheritance in Man (OMIM) database. Gene locations and HGNC gene symbols for the construction of artificial linkage intervals were extracted from Ensembl[20] using BioMart [21]. All the data used in this work was downloaded from the respective websites between April and June 2011. The exact file names and versions can be found on page A in the appendix.

2.1.1 Phenotype data

The mimMiner phenotype similarity matrix contains similarity scores for 5080 MIM phenotypes based on text mining of Medical Subject Headings (MEsH) terms in OMIM records. Phenotypes are represented as a vector of weighted and normalized feature terms from relevant OMIM records (using full text and clinical synopsis). The similarity score between two phenotypes is calculated as the cosine of the angle between the two feature vectors, resulting in scores in the range of [0,1]. For a more detailed description, we refer to the work of van Driel et al [17].

2.1.2 Gene expression data

The human gene expression data we used, was collected using whole-genome gene expression arrays that target 44775 human transcripts. Su et al built an extensive gene atlas using a panel of RNAs derived from 79 human tissues[19]. Their study represents one of the largest quantitative evaluations of gene expression on the protein-encoding transcriptome.

2.1.3 Protein-protein interaction data

HPRD is the most extensive manually curated database for human proteins. The list of binary interactions we used for this work contains only confirmed protein-protein interactions. There are other databases for protein-protein interactions, such as BioGrid, IntAct, BIND and MINT, some of which contain protein-protein interactions mapped to human proteins from model organisms or predicted by computer models. We did not use these networks, but we show how they could easily be integrated with our current data. Lehne & Schlitt recently wrote a review of human protein databases [22].

2.1.4 Phenotype-gene associations

Connections between phenotypes and related genes were extracted directly from the OMIM plain text file. The connections relevant for each test case are filtered on the fly such that only associations between phenotypes and genotypes that are present in the networks are retained. For the gene network based on PPI data, 1428

phenotypes are retained, for the gene network based on GE data, 1630 phenotypes are retained.

2.1.5 Artificial linkage interval

To calculate artificial linkage intervals, we extracted HUGO Gene Nomenclature Committee (HGNC) symbols, chromosome name and starting location on the chromosome from Ensembl using BioMart. For our experiments, we used the 100 nearest neighbors and the gene under benchmarking as artificial linkage interval. The distance between two genes was calculated as the difference between their starting point, without regard to whether the gene is on the plus or minus strand. If two genes are on different chromosomes, we set their distance to infinity.

2.2 Constructing phenotype and gene networks

In this work, we used one kind of phenotype network and two kinds of gene networks. The phenotype network is created based on the mimMiner phenotype similarity scores. We chose to separately evaluate two ways of constructing this network, one based on K-nearest-neighbors (KNN) and one based on thresholds. In the case of KNN, a phenotype was connected to the k most similar (i.e. highest scoring) neighbors with an edge weighted by the similarity score. As a small variation of this, we also constructed an undirected (symmetrical) version of each KNN network, where nodes a and b are connected if either a is among b's k nearest neighbors or vice versa. Again, the weight of an edge is given by the similarity score. In the case of a threshold, each phenotype was connected to all its neighbors whose similarity score was equal to or above the threshold. Unlike KNN networks, this kind of network may have nodes with degree 0.

The gene network based on gene expression data was constructed in the same way as the phenotype network (i.e. KNN and threshold). Edges were weighted according to the correlation coefficient between the expression of the two genes. If one gene was sampled by several markers, the values were averaged for each of the 158 tissues before calculating the correlation coefficient.

The gene network based on protein protein interactions is a network in which two genes are connected, if the proteins they encode for interact with each other. All edges have the same weight.

In order to apply the random walker algorithm, the two networks, namely the phenotype and the genotype network must be combined into one. We achieve this by connecting phenotypes to genes according to the gene-phenotype associations obtained from the OMIM database. The phenotype-gene associations are therefore represented by a bipartite graph in which each edge has weight 1.

For efficient handling in computers, the networks are represented by their adjacency matrix. Suppose $P_{m\times m}$, $G_{n\times n}$ and $B_{m\times n}$ represent the phenotype adjacency matrix, the gene adjacency matrix and the bipartite graph respectively, then the adjacency matrix M of the combined network can be represented as: $A = \begin{bmatrix} P & B \\ B^T & G \end{bmatrix}$

2.3 Random walker with restart

In order to generate a ranking from the combined network, we used a combined distance measure based on a random walker model. Random walker is a graph algorithm useful for many applications in which similarity or proximity to a set of seed nodes must be determined [23]. Köhler et al [8] were the first to propose using the random walker algorithm for disease gene prioritization, but it is quite similar to other algorithms also used for disease gene prioritization based on networks, such as diffusion kernels and network propagation [13].

The random walker algorithm simulates a random walk on the graph starting from a set of seed nodes and moving only to its immediate neighbors in each step. Similar to a Markov-chain, the transition probability of the random walker to each neighboring node is proportional to the weight of the node. At each step, the random walker restarts from a seed node with a certain probability. To obtain a ranking in the end, the nodes are sorted according to the average amount of time the random walker spent at each of them. This can be formulated mathematically in the following way:

Let p_0 be the seed vector, M the transition matrix of the graph and p_s the vector where the i-th element represents the probability of finding the random walker at node i after s steps. The probability vector at step s + 1 is then given by:

$$\mathbf{p}_{s+1} = (1-\gamma)M\mathbf{p}_s + \gamma\mathbf{p}_0 \tag{1}$$

Where γ is the restart probability. After a number of iterations the probability vector will reach a steady state $\mathbf{p}_\infty$. In practice, it is sufficient to stop iterating when the difference in L_1 space between $\mathbf{p}_s$ and $\mathbf{p}_{s+1}$ drops below a certain threshold (we used 10^{-10}). Due to rounding errors, the algorithm may never terminate if the threshold chosen is too small. $\mathbf{p}_\infty$ can be interpreted as a proximity vector respective to the seed nodes. The higher the value, the closer a node is to the seed nodes.

2.4 Random walk on the heterogeneous network

To apply the random walker algorithm to the heterogeneous network of genes and phenotypes, its adjacency matrix must be transformed into a stochastic matrix. The construction of the stochastic matrix presented here is based on the RWRH algorithm by Li & Patra [12].

From the previously constructed heterogeneous matrix Awe can construct the transition matrix

$$M = \begin{bmatrix} M_P & M_B \\ M_{B^T} & M_G \end{bmatrix}$$

Two additional parameters, λ and η are needed. λ represents the probability to jump from phenotype to gene network and vice versa and η is an optional parameter for weighting the importance of phenotype versus gene seed nodes in the initial vector. If λ is 0, genes and phenotypes will not be connected and will be ranked independently. Similarly, if η is larger than 0.5, the random walker will be more likely to (re-)start from the phenotype nodes. Not all genes are connected to

a phenotype and not all phenotypes are connected to a gene. This must be taken into account when constructing the transition matrix. The four sub-matrices of M are then constructed as follows.

The transition probability from phenotype p_i to gene g_j is given by:

$$(M_B)_{ij} = p(g_j|p_i) = \begin{cases} \lambda B_{ij} / \sum_j B_{ij}, & if \sum_j B_{ij} \neq 0 \\ 0, & otherwise \end{cases} \tag{2}$$

Conversely, the transition probability from gene g_i to phenotype p_j is given by:

$$(M_{B^T})_{ij} = p(p_j|g_i) = \begin{cases} \lambda B^T_{ij} / \sum_j B^T_{ij}, & if \sum_j B^T_{ij} \neq 0 \\ 0, & otherwise \end{cases} \tag{3}$$

The phenotype-phenotype transition matrix is defined by:

$$(M_P)_{ij} = p(p_j|p_i) = \begin{cases} 0 & if \sum_j P_{ij} = 0 \\ P_{ij} / \sum_j P_{ij}, & if \sum_j B_{ij} = 0 \\ (1-\lambda) P_{ij} / \sum_j P_{ij}, & otherwise \end{cases} \tag{4}$$

And similarly the gene-gene transition matrix is defined as:

$$(M_G)_{ij} = p(g_j|g_i) = \begin{cases} 0 & if \sum_j G_{ij} = 0 \\ G_{ij} / \sum_j G_{ij}, & if \sum_j B_{ji} = 0 \\ (1-\lambda) G_{ij} / \sum_j G_{ij}, & otherwise \end{cases} \tag{5}$$

Let $\mathbf{u}_0$ be the seed vector of the phenotype network and $\mathbf{v}_0$ the seed vector of the gene network. For the purpose of gene prioritization we assign equal weights to each seed gene in the gene and phenotype seed vector and scale them such that the sum of the weights is equal to 1 for both seed vectors. We then produce the seed vector for the heterogeneous network as $\mathbf{p}_0 = \begin{bmatrix} \eta \mathbf{u}_0 \\ (1-\eta)\mathbf{v}_0 \end{bmatrix}$. We can now apply the random walker algorithm as previously described. Phenotypes and genes will be ranked according to $\mathbf{u}_\infty$ and $\mathbf{v}_\infty$ given by $\mathbf{p}_\infty = \begin{bmatrix} \eta \mathbf{u}_\infty \\ (1-\eta)\mathbf{v}_\infty \end{bmatrix}$.

2.5 Fusing gene networks

Let A and B be the row-normalized transition matrices of the two networks to fuse. The fused matrix is then given by: $M = \alpha A + (1-\alpha)B$, where α is a parameter that defines the weight of each subnetwork. In practice this only works if the two matrices are of the same size and indexed by the same genes in the same order. When this is not the case, we can construct matrices S_A and S_B similar to permutation matrices, such that the fused matrix M will be given by

$$M = \alpha S_A A S_A^T + (1-\alpha) S_B B S_B^T \tag{6}$$

Given the set of genes U and V in the two separate matrices, the set of genes in the fused matrix is given by $W = U \cup V$. Let $f : U \to I_A$ and $g : W \to I_M$ be the bijective functions from gene name to matrix index of an input matrix and the fused matrix. The nonzero elements of the matrix S_A are then given by the following equation.

$$(S_A)_{ij} = 1 \quad \forall\, u \in U \text{ s.t. } f(u) = j \text{ and } g(u) = i \tag{7}$$

To adjust for cases in which a gene is only present in one of the networks, the fused matrix M must be row-normalized to become a transition matrix.

Obviously, this method can easily be extended to fuse more than two networks.

2.6 Validation methods

To evaluate our algorithm on different networks, we used the following two validation methods.

2.6.1 Leave-one-out cross-validation

In order to assess the accuracy of the ranking, we remove one direct link between a phenotype p_i and a causative gene g_j to subsequently test, whether the algorithm can recover this link. In practice this is done by removing B_{ij} from the bipartite graph and then ranking the genes based on the new transition matrix. If the algorithm ranks g_j first, we consider it a perfect ranking. This ranking can either be done relative to all other genes or relative to the genes in the artificial linkage interval. The genome-wide comparison simulates a ranking of genes for which no susceptible locus has been determined while the artificial-linkage comparison simulates diseases for which a linkage-interval is known.

2.6.2 Ab-initio cross-validation

Leave-one-out cross-validation only deletes one phenotype-gene link, but many phenotypes have more than one causative gene and many genes are involved in causing multiple phenotypes. In the dataset obtained from OMIM, one gene is involved in 1.56 phenotypes on average and one phenotype is associated with 2.88 known genes on average. Arguably, this makes recovering phenotype-gene links very easy for any algorithm. To simulate the case of phenotypes with no known causative genes and no susceptible chromosomal locus, we can perform whole-genome *ab-initio* prediction. To do so, we delete all the links between a phenotype p and its associated genes by setting the i-th row of B to zero and running the algorithm with the updated

transition matrix. We consider it a perfect ranking if one of the known disease genes is ranked first over all others in the genome.

2.7 Performance measures

We used the following performance measures in combination with the above mentioned methods to compare the performance of our algorithm for different networks.

2.7.1 ROC

The receiver operating characteristic, often used in signaling theory, can be applied to gene prioritization,too. Instead of true positive rate (TPR) and false positive rate (FPR), we plot the proportion of true causative genes below a threshold rank (TPR) versus the proportion of non-causative genes below the threshold (FPR). To compare different ROC curves, the area under the curve (AUC) is often used. The higher the value, the better the predictor. A perfect predictor will have an AUC of 1, while a random predictor will get an average value of 0.5.

2.7.2 Enrichment

Another way to measure performance is fold-enrichment. If a method ranks known disease genes in the top m% of all candidate genes in n% of the test cases, it is said to have n/m-fold enrichment on average. For instance, if a method ranks 50% of the known disease genes in the top 1%, it is said to have 50-fold enrichment. We use this measure to compare our method with other methods, where they provided no other indications.

We find that fold-enrichment scores can be misleading, because the score for the top 1% need not be equal to the fold-enrichment score for the top 50%. We therefore applied another common measure, which we call average enrichment. To calculate the average enrichment score, the average rank of all disease genes is divided by half the number of candidate genes. This enrichment factor is equivalent to the expected speedup for finding causative genes when investigating them in the order they were ranked versus investigating them in random order. If there are 100 candidate genes and the causative gene is ranked 10th on average, the enrichment is $50/10 = 5$. Testing genes in the order they were ranked would on average be 5 times faster than testing them in random order.

3 Implementation

This section provides an overview of the actual implementation and can serve as a rough guide to anyone wishing to reuse or read the code. The code is made public and copyright-free. Anyone may use it for any purpose of their liking.

3.1 Platform

We decided to use Python for the implementation of the algorithm and all other necessary codes for a number of reasons:

- Python is non-proprietary and open-source. Therefore everyone interested in reusing can do so without having to obtain licenses to run software written in Python.
- Python is pre-installed on Windows and most UNIX-based systems including Linux and MacOS, therefore making it very easy to use. Installing additional packages or even Python itself is extremely simple.
- Python is very easy to learn, making it an ideal choice for biologists. With Biopython, there is already a large code base available for many biology-related applications.
- With iPython, Python has a very powerful interactive shell for all common operating systems that makes running, testing and playing around with code very simple and intuitive.
- Python (as opposed to matlab®, mathematica®, or others) offers a complete environment for writing software (scientific computation, visualization, graph algorithms, image processing, serial drivers, multi-core processing, graphical user interfaces, web servers, etc.), which makes it easy to make an algorithm available through a web-interface or put it into a standalone program once it has been shown to work.
- Python allows programming in many kinds of paradigms (including procedural, object-oriented and functional), thereby making it easy for programmers in other language to understand and write Python code.
- Most importantly for this project, Python provides excellent packages for scientific computation and visualization (NumPy, SciPy and Matplotlib).

Because Python is a scripting language, it has certain performance issues when compared with compiled languages. However, if performance is a major concern using PyPy instead of CPython (the standard python interpreter) or PsyCO on top of CPython will in many cases greatly reduce the performance gap or even close it completely. Unfortunately, this does not always work, because PyPy does not support all of the Python packages which contain C modules (for example NumPy and SciPy). In this work, we implemented parts of the algorithm in parallel to increase speed as much as possible on systems that have multiple CPUs.

3.2 Packages

One focus of our code was to make it modular and as simple as possible to be reused and adapted for different prioritization algorithms.

We put all code in a package called nxpl (short for network explorer). It contains the following six sub-packages:

- **parse**:
  This package contains all functions necessary for parsing the input files such as plaintext HPRD and OMIM databases or Affymetrix gene expression data. Output consists of a matrix as well as an index for the rows in the matrix (for example the MIM number). This is done to ease conversion between different formats. The package also contains functions to create sparse or dense adjacency matrices for KNN or thresholded networks. For parsing GE data there is a special function that can be used in case the matrix of correlation coefficients is too large to keep in memory (in our example calculating the correlation coefficient matrix of 12851*12851 used more than 2 GB). If there is not enough space available to keep all correlation coefficients in memory, the function will store intermediate results in files and apply the threshold or KNN before creating the sparse matrix.
- **fuse**:
  This package is specific for prioritization algorithms that combine multiple networks. In our case, it contains the functions used for creating the heterogeneous network and its corresponding transition matrix.
- **neighbors**:
  This package contains functions to create the artificial linkage intervals used for validation.
- **solve**:
  This package contains the random walker algorithm.
- **validate**:
  This package contains the validation methods described in materials and methods. It performs leave-one-out cross-validation and *ab-initio* prediction.
- **visualize**:
  This package contains several functions for visualizing the results, including making ROC curves, density- and scatter-plots. It contains all the functions that were used to create the figures in the results section. All figures in this work were created with the functions contained in this package.

For convenience, an example script is also provided, which shows the use of the other packages. Given HPRD, GE and OMIM input files and parameters, it performs all the steps necessary to create the transition matrix and then does cross-validation checks before generating graphs as output. Scripts used to run tests and store results are also given for the sake of completeness, but are not as neat and tidy as the code in the nxpl package

To get the documentation for all the functions in the package, one can simply type pydoc nxpl.<name of package> in the terminal.

4 Results

In this section, we compare the characteristics of gene expression and phenotype networks constructed in different ways to quantify the impact of network construction on the quality of the rankings produced by the random-walker algorithm on the heterogeneous network (RWRH). We compare the performance of the random walker algorithm on different gene expression networks and a protein-protein interaction network in combination with different phenotype networks. Finally, we fuse a gene expression and phenotype network to create a larger network and compare the rankings produced with it to those produced with PPI and GE networks alone.

4.1 Network characteristics

The gene expression data contains samples for 33689 markers in 158 tissues. Because some genes contain several markers, the total number of genes sampled is only 12851. Where several samples existed for one gene, we decided to use the average of the expression of all markers for each tissue. In order to build a network with the gene expression data, some measure of gene proximity must be used. In our study, we used the absolute value of the standard correlation coefficient.

To build a network from the correlation coefficients, we applied a threshold and set all values in the matrix to zero, where the absolute value of the correlation was below the threshold. If the network has too few edges, it will be divided into many separate components, which has a negative impact on the quality of the rankings produced. On the other hand, if the network has too many edges, it becomes too large to be efficiently handled. Choosing the right threshold is therefore important. Figure 1 shows a density plot for the values in the correlation coefficient matrix, which can help making the decision of where to set the threshold.

Given the smoothness of the distribution, we decided to construct several networks with thresholds of 0.5,0.6,0.7,0.8 and 0.9 respectively. We also generated networks with smaller thresholds, but could not use them for efficiently ranking genes, because their adjacency matrices take up several gigabytes of memory each. Figure 2 shows the number of edges in the GE network for different thresholds. Our tests showed that GE networks need to be quite large in order to be fully connected (ie have only one connected component). To explore the influence of node connectivity on the results, we also constructed KNN (k-nearest neighbors) networks from the GE data. While the adjacency matrices of the GE network constructed with thresholds are all symmetrical, it should be noted that this is generally not the case for the KNN adjacency matrices.

Figure 3 shows a box plot for the number of connections per node for five different gene expression networks as well as for the protein-protein interaction gene network. Figure 4 shows a histogram of the number of edges per node for both networks. Neither the PPI network nor the GE-network are clearly scale free. The distribution of edges per node is almost uniform in the range of 50 to 100. The GE network has many nodes with few connections and its edge distribution is more similar to that of a scale-free network. However, like the PPI network, it also has a long flat tail. This observation hints, that networks constructed from correlation coefficients between the sampled markers in the gene expression need not have the scale-free property manifest in many biological networks [24].

The phenotype network was constructed from the mimMiner phenotype similar-

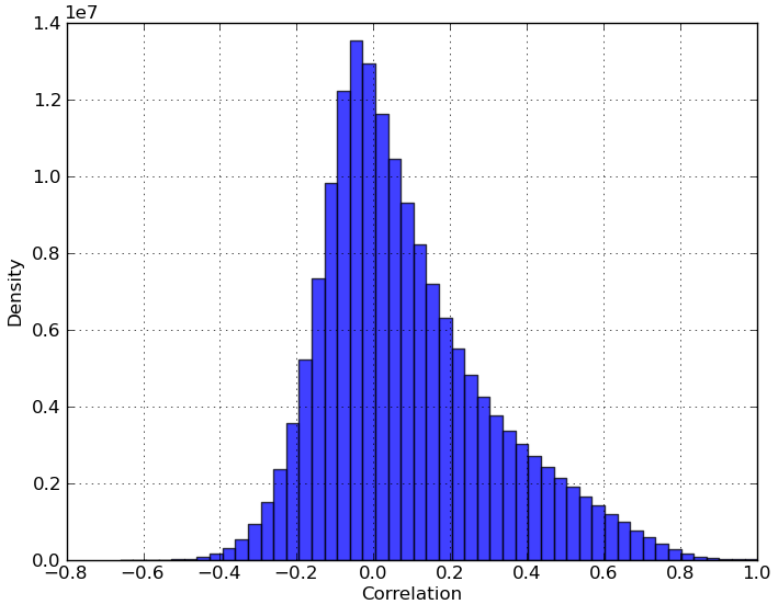

Figure 1: Distribution of the correlation coefficients for the expression of 12851 genes in 158 tissues

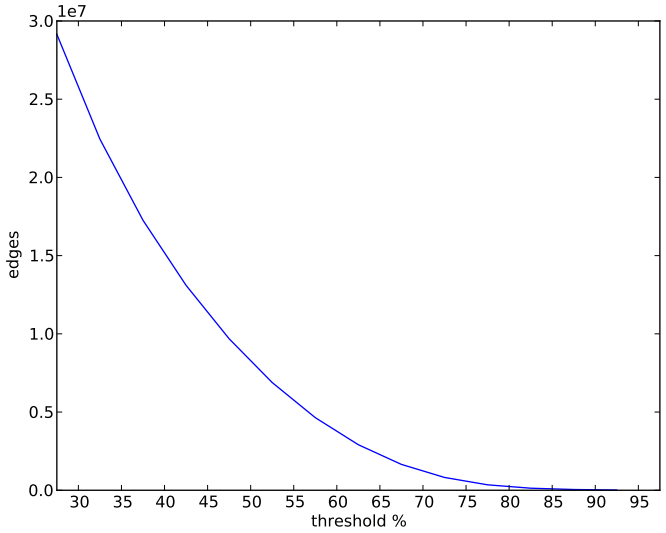

Figure 2: Total number of edges in GE network for different thresholds

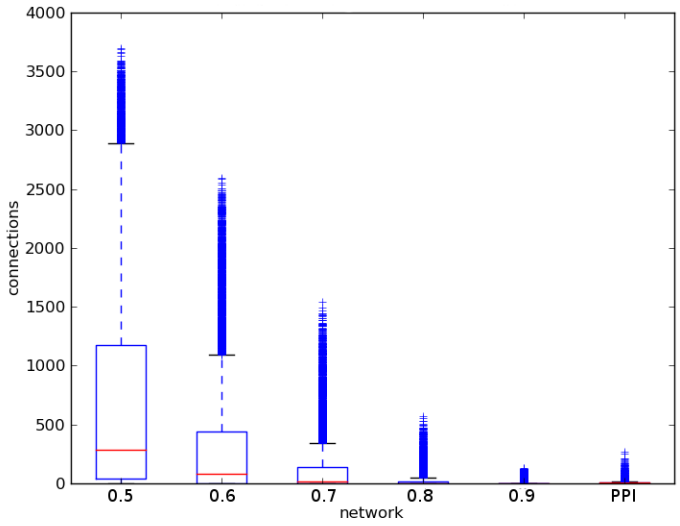

Figure 3: Box plot of the number of connections per node for gene expression networks with correlation threshold 0.5,0.6,0.7,0.8 and 0.9 and the PPI network

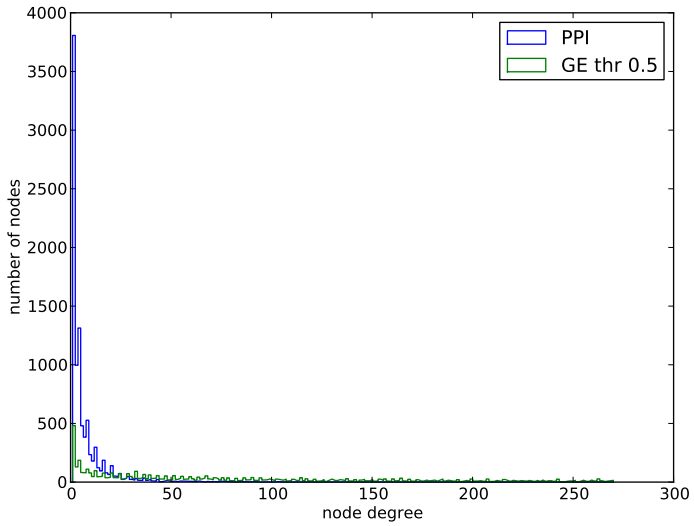

Figure 4: Histogram of the number of connections for the PPI network and the gene expression network with correlation threshold 0.5. The PPI network has 9607 nodes and the gene expression network has 12851 nodes.

ity data and includes 5080 phenotypes. We constructed different phenotype networks in the same fashion as the networks from GE data, that is with different thresholds and different k for KNN. Finally, we also constructed undirected (i.e. symmetrical) KNN networks for the phenotype data. The protein-protein interaction data we used was a binary interaction table that contained no metabolic rates or concentrations. Because of this, there is no basis for pruning nodes and the protein-protein interaction network is therefore always the same in our comparisons.

4.2 Performance of GE networks

To assess the usefulness of the networks with respect to disease gene prioritization, we performed leave-one out cross-validations for a set of 1630 known disease genes on an artificial linkage interval of 100 genes. The different measures used to assess performance are described in Materials and Methods. The numbers for whole-genome cross-validation correlate very well with those of the artificial linkage interval cross-validation, so we do not show any graphs for it here. The comparison tables can be found in the appendix.

For a useful candidate gene prioritization, it is preferable to include a maximum number of genes in the ranking. The GE network we constructed is significantly larger than the PPI network and thus better in this respect, but suffers from another drawback. Ranking is not possible for all genes, if the network is so sparse that some of them are not in the same connected component as any of the seed nodes. If this occurs, we assign no rank to the gene and call it a null-prediction. The smaller the number of edges in the network, the more likely it is, that this will happen. While null-predictions also occur for the PPI network, we found them to be much less common than for the GE networks. To assess when predictions are possible and when not, we plotted the number of perfect rankings (i.e. where the left-out causative gene is ranked first) versus the number of null predictions for different combinations of phenotype and gene networks.

Figures 5 and 6 show the influence of the threshold or KNN-parameter on the number of perfect rankings and null predictions for different phenotype and gene networks. Figure 7 shows the AUC scores for the same networks. We compare only the performance of symmetrical KNN phenotype matrices, because they produced much better rankings than the thresholded version. Compared with the non-symmetrical phenotype networks, the performance is similar, but slightly better.

These plots show that while the number of cases in which the left out gene is ranked first stay perfectly constant for different KNN phenotype networks, the number of null-prediction increases linearly with the sparsity of the network. Sparsity of the gene networks on the other hand affects both the number of perfect rankings and the number of null-predictions. This suggests that the gene network and not the phenotype network is responsible for the clustering of the heterogeneous network into several smaller connected components. As the figures show, the more edges a gene expression network has, the better its performance. The influence of the threshold on the number of perfect rankings is much smaller than the effect on the number of null-predictions. In the extreme case of the phenotype network with threshold 0.6 and the GE network of a threshold of above 0.9 (not shown in figures), the gene is therefore either ranked first or not ranked at all. This suggests that the dataset used for validation has a certain bias to well studied genes and easily discovered disease to gene relationships. Evidence is in this case firmly established

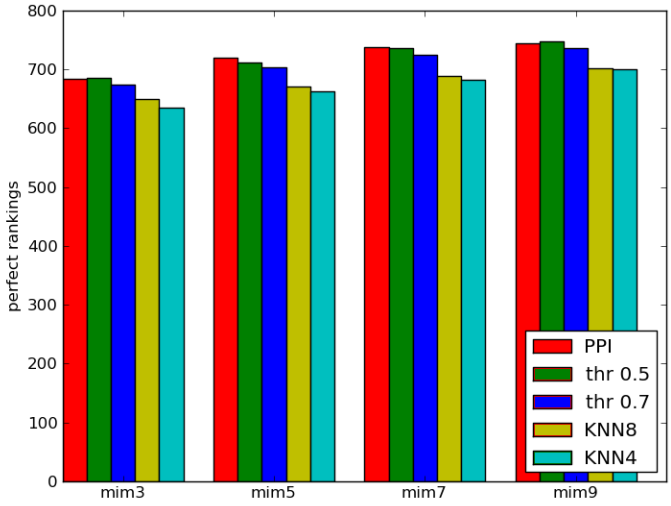

Figure 5: Comparison of the number of perfect rankings between different phenotype and gene networks

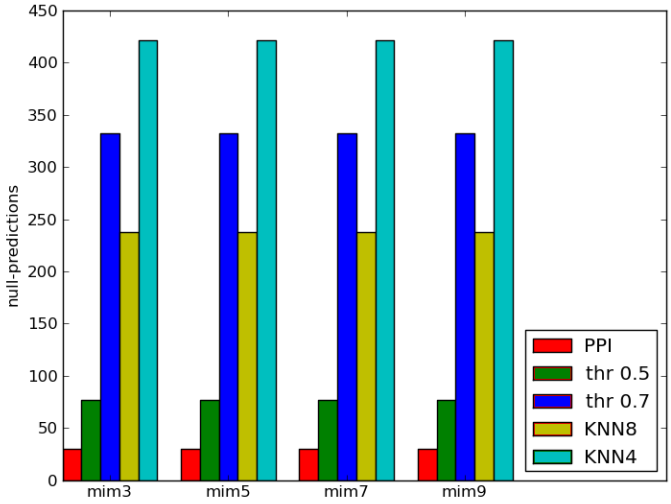

Figure 6: Comparison of the number of null-predictions between different phenotype and gene networks

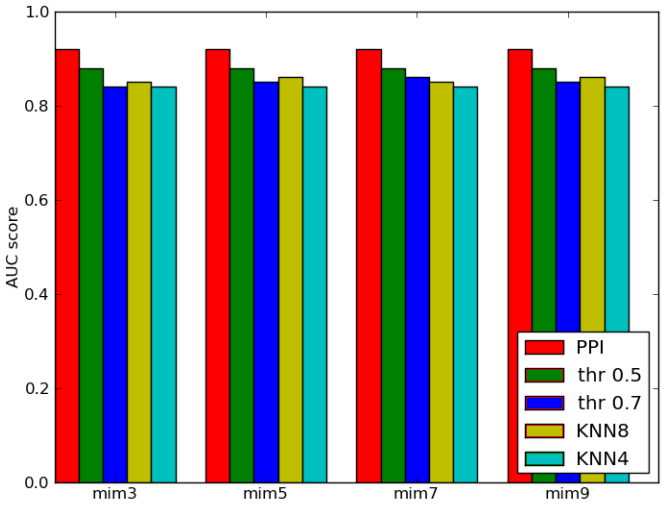

Figure 7: AUC scores for different combinations of phenotype and gene networks)

through several connections between the phenotype and neighboring genes. If we ignore the null-predictions, the areas under the ROC curve for the GE network with the thresholds plotted would be extremely close to 1 (0.97) This shows that depending on the measure applied, the algorithm can appear to be almost perfect, even if it fails to rank a majority of the genes. There is however a way to make use of this phenomenon as a feature: To predict novel disease genes, the algorithm could be run many times, lowering the threshold at each turn until a ranking is produced. Figure 8 shows the ROC curve when null-predictions are ignored versus when they are assigned a random rank.

For the number of perfect rankings, all networks show comparable performance with differences of less than 10%. The PPI network is always among the best and only sometimes beaten by the much denser gene expression network with threshold 0.5. But these numbers should be interpreted with care, because they contain no information about genes not ranked in the first place. As the comparison of AUC values in figure 7 shows, the number of perfect rankings is not a reliable indicator for overall performance. In terms of AUC score, the protein-protein interaction network constantly beats the gene expression networks. Again, the density of the phenotype network has only a negligible effect.

For the remainder of this section, we used the phenotype network constructed with KNN=9 and an RWRH ranking on an artificial linkage interval of 100 genes is used to create all the plots. Tables for all the different performance measures of all the phenotype networks in combination with the GE and PPI networks, including whole-genome rankings, can be found on page A in the Appendix.

ROC curves for different thresholds and KNN values for the genotype network are shown in figure 9. In this case, a random rank was assigned for null-predictions to make the ROC curve smooth.

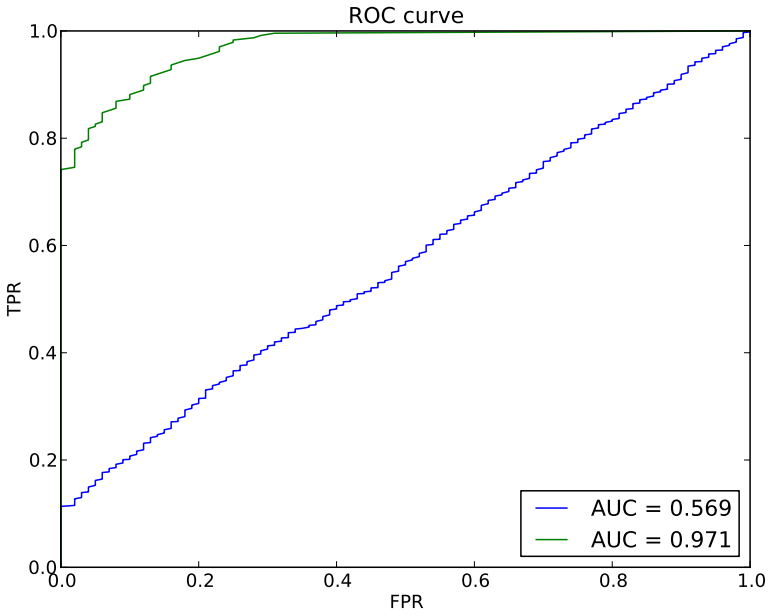

Figure 8: Artificial linkage interval ROC curves for RWRH ranking on the phenotype network with threshold 0.6 and the gene network with threshold 0.9, once ignoring null-predictions (green) and once assigning random rank to null-predictions (blue).

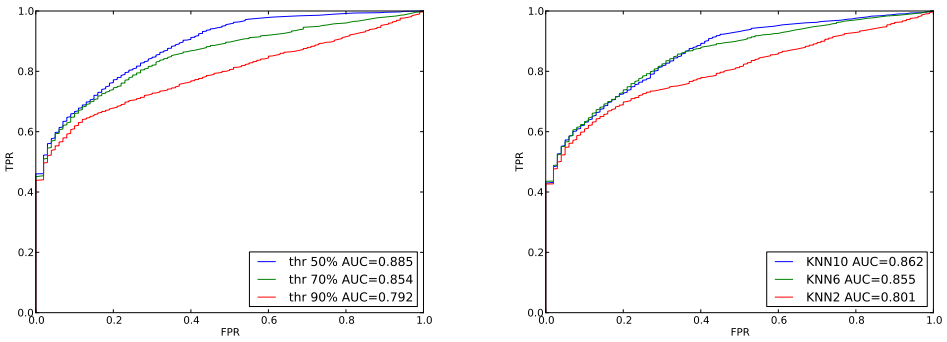

Figure 9: Artificial linkage interval ROC curves for RWRH ranking using gene expression networks with different thresholds and KNN values in combination with a symmetrical 9-KNN phenotype network.

The mean enrichment values for different phenotype networks and rankings based on the PPI network varied between 10 and 24 while the same values for GE networks were between 8 and 20. Since the artificial linkage interval contains 100 genes, the enrichment is 50/rank(gene). If the gene is ranked first, the enrichment is 50. In cases were the gene could not successfully be ranked, we used an enrichment value of 1 (=average rank).

It is reasonable to assume, that fully connected networks will produce the best results, because they would make use of all the available data. While the evidence indicates, that a lower threshold leads to better performance, we could not make use of this due to memory constraints. As the number of edges in the network grows, so does computation time for multiplication (complexity of sparse matrix multiplication depends linearly on the number of nonzero values). Fortunately, the data in the tables on page A indicate that while additional edges increase the quality of the rankings, the effect levels off after a certain point. It can thus be expected that even using a fully connected phenotype network and gene expression network would lead to only marginally better results while incurring the cost of polynomially increasing space and time consumption.

After applying many different comparison methods, we observed that the values were always highly correlated, i.e. if combination X of matrices has a higher AUC score than combination Y, it also has a higher mean enrichment score. This is also true for cross-validation on the artificial linkage interval versus cross-validation on the whole genome. As already mentioned, the relationship between null predictions and perfect rankings is only very weak and the number of null-predictions cannot be used to predict the number of perfect rankings and vice versa. Because of this, we used only the best combinations of networks for the comparisons in the following sections: the symmetrical phenotype network with KNN=9, the genotype network with threshold 0.5 and the standard PPI network.

4.3 Fused network

In order to get rankings for a maximum number of genes and a maximum number of phenotypes, we constructed a combined matrix from the best GE network and the PPI network. The new network contains 14612 genes, compared to 9607 for the PPI network alone and 12851 for the GE network alone. Figure 10 shows the ROC curve for the combined network with the two single networks.

While it could be speculated that the value of α, the parameter for weighing the contribution of the two networks in the final network, has an impact on the performance, we did not find this to be the case: We ran trials for different values of α ranging from 0.1 to 0.9 in steps of 0.1. The best performance was obtained at α=0.3, but the difference between the highest and the lowest AUC values was smaller than 0.01

We conducted a leave-one-out cross-validation between the fused network, the GE network and the PPI network to find out whether the larger size (and thus a higher possible worst rank and added noise) of the new network significantly lowers its performance. To make the comparison as unbiased as possible, we performed leave-one-out cross-validation on a subset of genes that were present in all three networks. Figures 11 and 12 show scatter-plots where the ranks of each plotted against the other for the leave-one-out cross-validation on the random linkage interval. Equal ranks lie on the diagonal. The fitted linear regression line shows that the

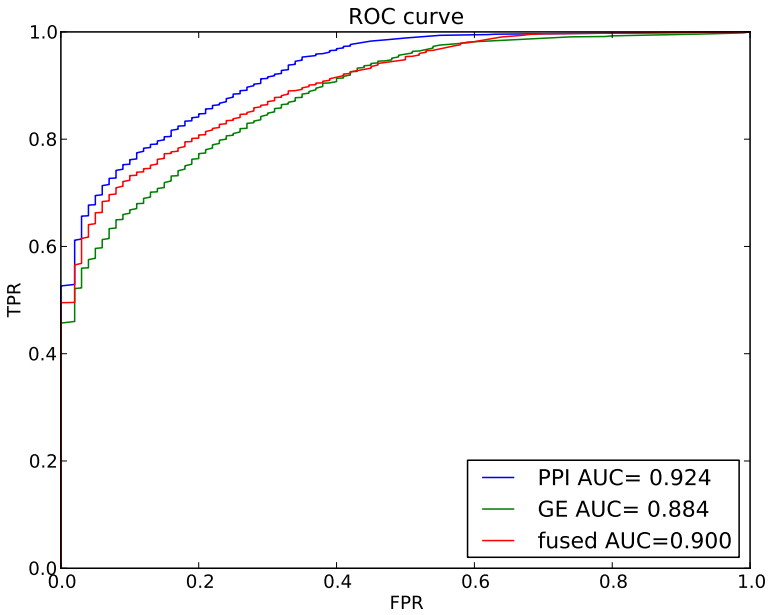

Figure 10: Comparison of ROC curves for RWRH leave-one out cross-validation rankings with PPI network, GE network with threshold 0.5 and the two networks fused together (α=0.3). There were 1717,1630 and 1428 gene-phenotype connections for the fused network, the GE network and the PPI network respectively.

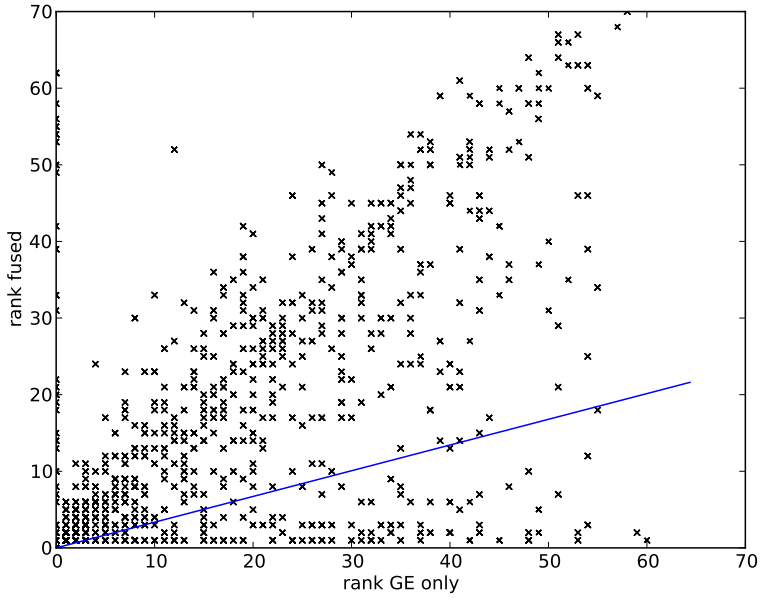

Figure 11: Scatterplot of rank of fused network plotted against rank of GE network only for leave-one-out cross validation (1630 connections) on an artificial linkage interval of 100 genes. Values on the diagonal mean equal rank was assigned. The blue line shows the best linear fit (with minimal square error), indicating that the performance of RWRH on the fused network is superior to that of RWRH on the gene expression network alone.

performance of the fused network is not worse, but actually superior to that of each of the separate networks, despite the fact that it ranks 20% and 50% more genes respectively.

The AUC values for RWRH with each of the three different networks can be seen in figure 13. This time, all possible connections are used (1717, 1428 and 1630 gene-phenotype connections for the fused network, the PPI network and the GE network respectively). The values indicate that the fused network is always better than the GE network alone. On the artificial linkage interaval and for *ab-initio* cross validation, the fused network and the PPI network show almost identical performance. For whole-genome cross-validation the fused network is better than the PPI and GE networks. This could be due to the fact that the fused network is more complete in terms of connections between related genes and can thus produce more accurate ranks even when the gene is not in close proximity to the phenotype. The ROC curves for the whole-genome cross-validation (figure 14) seem to support this hypothesis.

The rest of the comparisons are not shown here for the sake of brevity. A summary can be found in the tables on page A of the appendix.

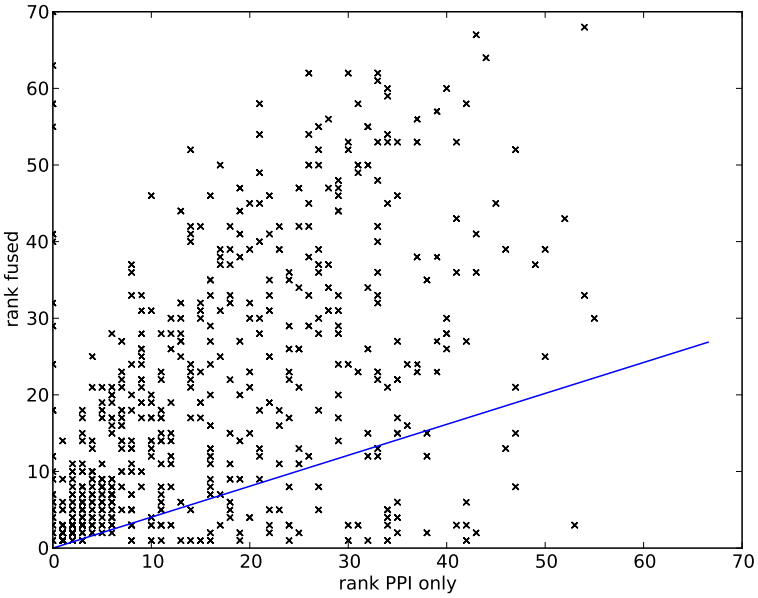

Figure 12: Scatterplot of rank of fused network plotted against rank of PPI network only for leave-one-out cross validation (1428 connections) on artificial linkage interval of 100 genes. Values on the diagonal mean equal rank was assigned. The blue line shows the best linear fit (minimal square error), indicating that the performance of RWRH on the fused network is superior to that of RWRH on the protein-protein interaction network alone.

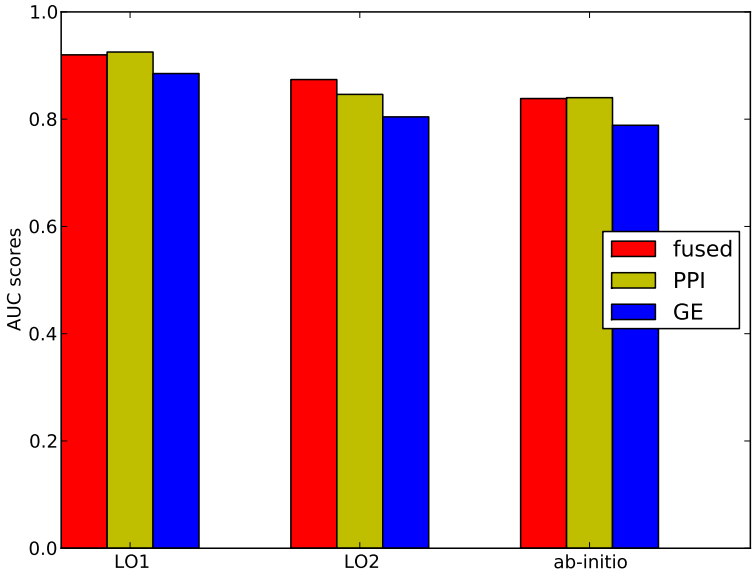

Figure 13: AUC scores for the three networks for leave-one-out cross validation on artificial linkage interval, whole-genome leave-one-out cross-validation and *ab-initio* cross-validation. The number of validation cases were 1717,1428 and 1630 for the three different networks.

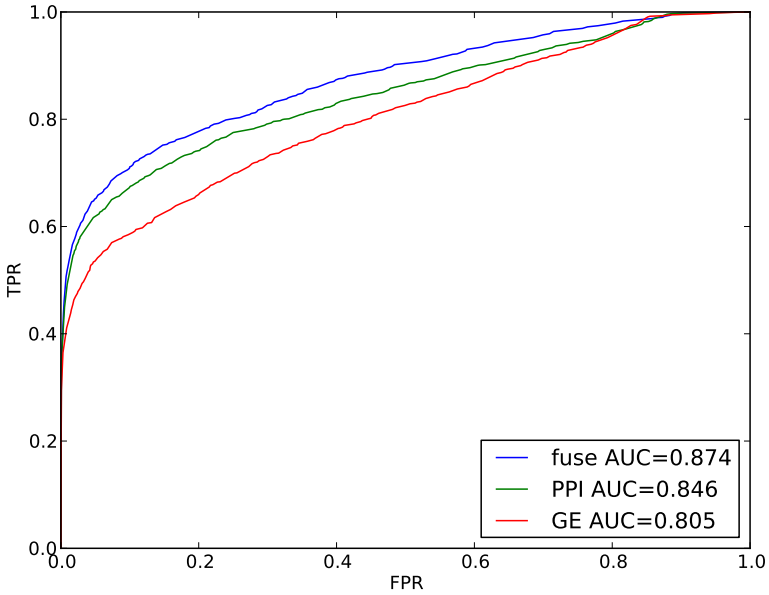

Figure 14: ROC curves for the three networks for whole-genome leave-one-out cross-validation. There validation set included 1717,1630 and 1428 gene-phenotype connections for the fused network, the GE network and the PPI network respectively.

4.4 Comparison of fused network with other methods

Various other methods for disease gene prioritization have been proposed, but it is difficult to compare their performance methods, because they often use different data sets and different validation methods. The following part, in which we try to compare the performance of RWRH with the fused network to that of other methods, should therefore be taken with a grain of salt.

The algorithm used in this work is based on the random walk with restart on a heterogeneous network (RWRH) proposed by Li & Patra [12], who based it on the random walk with restart algorithm from Köhler et al [8]. The comparison with RWRH is implicit, since they used the same phenotype data as well as a subset of the protein-protein connections from HPRD which we used. Their network contained 8919 proteins while our network contains 9607. One could expect that the addition of more protein protein connections increases the accuracy of their algorithm, but the opposite seems to be the case. For RWRH, the number of genes ranked in first place is 814 and 245 for leave-one-out cross-validation on the artificial linkage interval and whole genome respectively and 201 for *ab-initio*. These numbers are surprising, because the best scores we produced for the combination of a symmetrical phenotype network and a PPI network were only 744, 209 and 191. It could be, that the addition of the 688 proteins somehow lowered the performance, but this is unlikely. However, because they did not mention how they dealt with null-predictions, we believe it is more likely, that this is the explanation for the discrepancy. If the random walker does not reach any gene, all genes will have equal score and thus the equal rank of 1. In our trials, we did not use the 50 null-predictions for counting the number of perfect rankings. If they are taken into account, the difference between our numbers and those of Li & Patra are reduced to 20 in the case of artificial linkage interval rankings and even reversed to -14 for the whole-genome ranking. There is still a difference, but this can be explained with the fact that we used more interactions and a phenotype matrix with more connections. To validate the result of Li & Patra, we would have to repeat the experiments with their input data. The *ab-initio* validation did not produce any null-rankings. We can confirm the result that the influence of the parameters η, λ and γ on the performance of the algorithm is minimal. For our figures and tables, we therefore used the same values as they did, i.e 0.7 for λ and 0.5 for η and γ

Wu et al also used phenotype similarity, protein-protein interaction networks and known gene-phenotype connections as input for their algorithm. They generated the rankings based on a regression model for proximity profiles of diseases and genes [9]. In their paper, they list the number of perfect predictions as well as the fold-enrichment for the first 1%. The fold-enrichment of our fused network is 51.5, compared to 53.5 for CIPHER, which is almost identical. For the whole-genome prediction, the value for CIPHER is 954 while for our method it is 1873, which is almost twice as much.

Other methods used for prioritization include ENDEAVOUR [11], PRINCE [13] and RWR [8]. We are not comparing our results with theirs, because their input data as well as their validation sets are radically different from ours, making a meaningful comparison very difficult. In the discussion, we make a few suggestions on possible improvements that would make it easier to compare methods in the future.

5 Comparison and reproducibility of results

Comparisons and performance evaluations are a big problem for current algorithms. Most papers published on disease gene prioritization involve some sort of comparison with previous methods, but because many of the methods are hand-crafted for a specific set of manually curated data, it is difficult to compare them with one another. CIPHER for example uses a phenotype network containing 5080 phenotypes and a PPI network containing 8919 proteins, while ENDEAVOUR uses ontologies (GO, KEGG, ...), microarray data, pathways and sequence similarity as data sources. Each method alone ranks several thousand genes and was evaluated on several hundred of them, but the overlapping test set only contains 80 genes, not enough to prove that one method is significantly better than the other.

One problem that makes comparing different methods difficult, is that there is no obvious and unbiased performance measurement. The first problem with measuring performance is due to the fact that those methods that rely on known gene-phenotype or gene-disease associations to make their ranking cannot simply be tested on known disease genes. As we have seen, a common way to forgo this issue is leave-one-out cross validation. However, since the algorithms are based on the modular nature of human diseases, it is not surprising that they will in a majority of cases correctly rank the left out connection at the top. In that case, we are in fact just measuring how modular the network really is.

In the data we retrieved from OMIM, one phenotype was related to 2.88 genes on average and each known disease gene was involved in 1.56 phenotypes on average. In case there are multiple connections from one gene or phenotype, leaving out only one of them will make prediction quite easy. A possible solution is to ignore all phenotypes sharing the same benchmarked disease gene, but this only works in leave-one-out cross-validation [9], because for *ab-initio* prioritization the modular nature of the network would be destroyed. Another way of evaluating performance while keeping a minimal bias is to test the performance on recently discovered disease genes. This method was used in ENDEAVOUR and RWR. Considering only recently discovered disease genes can reduce the literature and selective bias (disease genes are likely to be better studied than others), but the number of genes in the sample is usually very low. ENDEAVOUR for example was assessed on only 16 genes whose association to disease had been recently discovered [11]. But even for this kind of evaluation there should be concern that a significant bias remains, because new disease genes are more likely to be discovered if they are well-researched and present in many datasets.

Even comparing closely related methods is rendered difficult through the fact that the source code is almost never made public or shared and that the exact input data cannot be determined from the information available. Where the code is made public, it usually contains only the core algorithm and a possible benchmark, but not the code necessary for preprocessing the data set. Since it is usually the preprocessing that is most complex and error prone, not much is gained. Making the algorithm accessible through a website is praise-worthy and certainly useful for inspecting results, but it is not suitable for direct comparison with other methods, since the input data cannot be changed by the user. A clear drawback of the non-availability of code is that unless the authors keep developing the software, no bugs will ever be discovered or corrected. We believe that requiring authors to always publish code would benefit everyone by making independent verification and

comparison possible. A probable side-effect is that the code would become more structured and readable, which is very likely to help the author reduce the number of bugs even before publication.

As partial solutions for the problems mentioned above, especially concerning comparison and reproducibility, we propose the following:

1. Publication of algorithms should always include publication of all associated code and clear identification of the data sources used. If previously non-public data sources are used, they should be made available too. The code included should be able to do all the processing from parsing the original data source to validation of results in order to guarantee reproducibility and enable detection of bugs. This would further allow interested researchers to adapt the method to their needs and use the input data they want.
2. A general framework for implementing disease gene prioritization software and standardized output formats to make using, testing and combining different algorithms as simple as possible would be very useful. It is especially important to have an interface which makes interpretation easy for biologists, because they are the ones that could benefit most from prioritization methods.
3. There should be a critical and generally agreed upon assessment standard for prioritization methods, similar to the CAGI community project for the prediction of phenotypic impacts of genetic variations [25]. This is somewhat more difficult to do for prioritization methods, but it is possible to evaluate performance of algorithms with input data published before the discovery of a number of test genes. Instead of delaying the assessment for several years, older datasets could be used. Most importantly, criteria for performance evaluation should be defined before results are available not chosen for convenience afterwards.

It might be difficult to get researchers to participate, but we believe that some steps in this direction are necessary, if there is sufficient continued interest in disease gene prioritization.

6 Conclusion

In comparison with protein-protein interaction networks used by several prioritization methods, gene expression data is relatively easy to generate and available for a great number of markers on the whole genome. The fact that our method using gene expression data produces rankings that are as good or even better than those of methods using a large number of manually curated datasets, is an extremely promising result. We have successfully demonstrated the excellent performance of a random walker algorithm on the combination of a protein-protein interaction network with a GE-network, but the true strength of the proposed method lies in the simplicity of combining data sources. Our fusion method bundles the power of prioritization based on the modular nature of diseases with the high throughput of gene expression studies. If our method is always as robust to changes in parameters as in the example we showed, it could prove to be an extremely useful tool for gene prioritization, making it possible to give a meaningful rank even to less well-studied genes which other methods do not include so far.

In the future, our method could be extended by evaluating and adding more gene expression data to eventually cover the whole genome. With the code base we provide, this process can easily be automated. Fusing networks is of course not limited to genes, but can also be applied to phenotypes.

Although we evaluated our method on several validation sets and proposed new ideas for how different methods could be compared, we believe that the usefulness of prioritization methods should be measured not on how they pass validation, but on how they are actually used for the discovery of disease genes by other scientists. Because usefulness for real-world applications is the real measure for algorithms in bioinformatics, extra care should be put into making it easier for researchers to get the information they really want. After all, the final goal is not to rank candidate genes, but to understand the mechanisms of disease and eventually discover a cure.

7 Acknowledgements

The research for this thesis was conducted at the Bioinformatics Division of the Tsinghua National Laboratory for Information Science and Technology (TNLIST) in Beijing, China, under the joint supervision of Professor Rui Jiang of Tsinghua University and Professor Bernard Moret of EPFL.

I thank Rui Jiang for his very kind support and supervision during the project. Many thanks also to Bernard Moret, due to whose efforts I was able to spend 6 months at Tsinghua University for research and writing this thesis. Furthermore, I would also like to thank my friends and colleagues for giving me critical input on the thesis and making my stay at Tsinghua University a great experience overall.

References

[1] Hamosh A, Scott AF, Amberger J, Bocchini C, Valle D, McKusic VA (2002), Online Mendelian inheritance in man (OMIM), a knowledgebase of human genes and genetic disorders, *Nucleic Acids Res.* 30, 52-55

[2] Botstein D, Risch N (2003), Discovering genotypes underlying human phenotypes:past successes for Mendelian disease, future approaches for complex disease, *Nat Genet*, 33(Suppl)228-237

[3] Glazier AM, Nadeau JH, Aitman TJ (2002), Finding genes that underlie complex traits, *Science*, 298(5692):2345-2349

[4] Lander ES, Schork NJ (1994), Genetic dissection of complex traits, *Science*, 256(5181):2037-2048

[5] Perez-Iratxeta C, Bork P, Andrade MA (2002), Association of genes to genetically inherited diseases using data mining, *Nat. Genet.* 31, 316-319

[6] Turner F et al (2003), POCUS: Mining genomic sequence annotation to predict disease genes, *Genome Biol*, 4,R75

[7] Adie EA et al (2006), SUSPECTS: Enabling fast and effective prioritization of positional candidates, *Bioinformatics*, 22, 773-774

[8] Köhler S, Bauer S, Horn D Robinson PN (2008), Walking the interactome for prioritization of candidate disease genes,*American journal of human genetics* 82: 949-958

[9] Wu X, Jiang R, Zhang MQ, Li S (2008), Network-based global inference of human disease genes, *Mol Syst Biol* 4:189

[10] Trabolusi EI et al (2006), Lumpers or splitters? The role of role of molecular diagnostics in Leber congenital amaurosis, *Ophtalmic Genet*, 27,113-115

[11] Aerts S et al (2006), Gene prioritization through genomic data fusion, *Nature Biotechnology* vol 24 no 5 p 537-544

[12] Li Y, Patra JC (2010), Genome-wide inferring gene-phenotype relationship by walking on the heterogeneous network, *bioinformatics* vol. 26 no 9 p 1229 - 1224

[13] Vanunu O, Magger O, Ruppin E, Shlomi T, Sharan R (2010), Associating Genes and Protein Complexes with Disease via Network Propagation, *PLoS Comp Biol* 6(1): e1000641

[14] Suthram S, Beyer A, Karp RM, Eldar Y, Ideker T (2008), eQED: an efficient method for interpreting eQTL associations using protein networks, *Mol Sysy Biol* 4:162

[15] Zhang W, Sun F, Jiang R (2011), Integrating multiple protein-protein interaction networks to prioritize disease genes: a Bayesian regression approach, *BMC Bioinformatics* 12(Suppl 1):S11

[16] Oti M, Brunner HG (2007), The modular nature of genetic diseases, *Clin Genet* 71:1-11

[17] van Driel MA, Bruggeman J, Vriend G, Brunner HG, Leunissen JA (2006), A text-mining analysis of the human phenome, /textitEur J Hum Genet, 14(5):535-542

[18] Keshava Prasad TS, Goel R,Kandasamy K, Keerthikumar S, et al (2009), Human Protein Reference Database-2009 update,*Nucleic Acids Res*,37:D767-772

[19] Su AI, Wiltshire T, Batalov S, Lapp H, Ching KA, Block D, Zhang J et al (2004), A gene atlas of the mouse and human protein-encoding transcriptomes textitProc Natl Acad Sci USA 101, 6062-6067

[20] Hubbard TJP, Aken BL, Beal K, Ballester B, Caccamo M, Chen Y, et al (2007), Ensembl 2007, *Nucleic Acids Res.* 35:D610-617

[21] Kasprzyk A, Keefe D, Smedley D, London D, Spooner W, et al (2004), EnsMart: a generic system for fast and flexible access to biological data, *Genome res* 14:160-169

[22] Lehne B, Schlitt T (2009), Protein-protein interaction databases: keeping up with growing interactomes, *Human Genomics*3:291-297

[23] Can T, Camolgu O, Singh AK (2005), Analysis of protein-protein interaction networks using random walks, *BIOKDD 05*: 61-68

[24] Khanin R and Wit E, (2006), textitJournal of Computational Biology,13(3): 810-818. doi:10.1089/cmb.2006.13.810

[25] http://genomeinterpretation.org/

8 Appendix

8.1 Input files

GE data of Su et al was downloaded from http://biogps.gnf.org/downloads/ filename:GNFH1data.xls (GEO code: GSE1133) and AffyU133Aannotation.txt

PPI data was downloaded from HPRD, filename: HPRD_Release9_041310.tar.gz BINARY_PROTEIN_PROTEIN_INTERACTIONS.txt

The OMIM database was downloaded in plaintext format, filename: morbidmap, last version downloaded on 9.6.2011

8.2 Tables of matrix comparisons

The tables on the following pages show a selection of the large scale comparison study we conducted.

Table 1: AUC values for leave-one out cross-validation on an artificial linkage interval of 100 genes

	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN10	KNN2	KNN4	KNN6	KNN8
KNN1	0.741	0.72	0.705	0.667	0.639	0.597	0.715	0.568	0.633	0.676	0.689
KNN3	0.888	0.832	0.824	0.804	0.781	0.716	0.819	0.732	0.774	0.8	0.813
KNN5	0.906	0.858	0.847	0.827	0.808	0.765	0.844	0.781	0.822	0.829	0.831
KNN7	0.913	0.867	0.86	0.838	0.81	0.78	0.85	0.796	0.819	0.839	0.847
KNN9	0.916	0.872	0.859	0.843	0.818	0.777	0.855	0.808	0.825	0.844	0.85
KNN_sy1	0.837	0.792	0.785	0.76	0.733	0.665	0.782	0.668	0.744	0.766	0.784
KNN_sy3	0.919	0.875	0.867	0.844	0.826	0.781	0.857	0.808	0.835	0.84	0.849
KNN_sy5	0.922	0.882	0.869	0.852	0.823	0.789	0.863	0.801	0.835	0.851	0.858
KNN_sy7	0.923	0.882	0.87	0.856	0.822	0.785	0.863	0.804	0.837	0.846	0.854
KNN_sy9	0.924	0.882	0.879	0.854	0.833	0.786	0.868	0.8	0.839	0.85	0.857
thr40	0.92	0.88	0.87	0.853	0.825	0.787	0.859	0.798	0.823	0.849	0.856
thr50	0.845	0.806	0.796	0.781	0.758	0.729	0.792	0.735	0.769	0.786	0.787
thr60	0.699	0.68	0.678	0.658	0.637	0.615	0.666	0.611	0.639	0.66	0.659
thr70	0.608	0.582	0.579	0.577	0.557	0.552	0.599	0.549	0.571	0.574	0.57
thr80	0.545	0.546	0.545	0.53	0.53	0.528	0.536	0.525	0.523	0.524	0.535

B

Table 2: null-prediction counts for leave-one out cross-validation on an artificial linkage interval of 100 genes

	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN10	KNN2	KNN4	KNN6	KNN8
KNN1	569	656	777	958	1157	1335	733	1384	1158	935	808
KNN3	61	129	254	417	615	890	257	813	572	426	319
KNN5	40	95	215	367	536	733	213	630	465	351	267
KNN7	32	82	197	342	507	694	196	592	437	327	248
KNN9	32	80	193	337	498	677	192	582	427	318	242
KNN_sy1	232	283	401	576	767	1090	379	1068	710	547	438
KNN_sy3	30	77	189	332	492	669	188	573	421	313	238
KNN_sy5	30	77	189	332	492	669	188	573	421	313	238
KNN_sy7	30	77	189	332	492	669	188	573	421	313	238
KNN_sy9	30	77	189	332	492	669	188	573	421	313	238
thr40	44	90	201	343	500	676	202	583	431	327	251
thr50	278	358	462	588	730	876	460	809	669	569	506
thr60	719	874	944	1031	1140	1240	946	1233	1108	1037	981
thr70	1042	1233	1280	1329	1394	1445	1278	1458	1400	1348	1301
thr80	1233	1440	1463	1495	1536	1551	1467	1555	1536	1505	1483

Table 3: Perfect ranking counts for leave-one out cross-validation on an artificial linkage interval of 100 genes

ones	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN10	KNN2	KNN4	KNN6	KNN8
KNN1	354	355	337	323	308	265	322	236	275	303	321
KNN3	584	559	556	539	546	518	520	480	509	517	510
KNN5	648	623	623	622	619	610	578	562	573	582	585
KNN7	683	670	662	657	665	641	620	614	613	618	622
KNN9	708	695	686	676	684	672	651	634	637	649	650
KNN_sy1	481	474	459	454	451	419	439	373	406	426	435
KNN_sy3	683	686	679	674	683	673	642	637	634	652	650
KNN_sy5	720	712	709	704	708	681	674	657	662	668	671
KNN_sy7	737	736	730	724	716	693	691	674	682	685	688
KNN_sy9	744	748	735	736	729	705	704	689	700	706	701
thr40	732	746	740	740	726	697	710	682	697	705	700
thr50	594	606	587	578	579	562	559	543	548	554	561
thr60	366	363	363	351	333	324	337	311	326	336	336
thr70	196	191	188	182	176	172	181	161	161	169	179
thr80	85	84	86	83	80	77	78	72	73	77	79

Table 4: Average enrichment scores for leave-one out cross-validation on an artificial linkage interval of 100 genes

	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN10	KNN2	KNN4	KNN6	KNN8
KNN1	1.869	1.702	1.637	1.513	1.343	1.242	1.713	1.177	1.359	1.539	1.647
KNN3	4.344	2.937	2.714	2.518	2.22	1.768	2.831	1.856	2.243	2.445	2.678
KNN5	5.068	3.399	3.124	2.915	2.563	2.079	3.071	2.402	2.709	2.883	2.992
KNN7	5.43	3.733	3.382	2.944	2.558	2.159	3.221	2.344	2.797	3.088	3.18
KNN9	5.49	3.795	3.422	3.1	2.652	2.3	3.251	2.512	2.951	3.004	3.267
KNN_sy1	2.978	2.391	2.283	1.995	1.876	1.486	2.273	1.527	2	2.083	2.219
KNN_sy3	5.783	3.739	3.525	3.111	2.743	2.334	3.362	2.45	2.848	3.195	3.316
KNN_sy5	5.981	3.982	3.762	3.338	2.802	2.314	3.519	2.459	2.85	3.174	3.409
KNN_sy7	6.011	4.091	3.781	3.265	2.843	2.275	3.524	2.529	2.964	3.352	3.48
KNN_sy9	6.132	4.151	3.856	3.242	2.906	2.37	3.471	2.519	3.031	3.297	3.533
thr40	5.799	4.113	3.704	3.177	2.703	2.272	3.471	2.499	2.803	3.157	3.295
thr50	3.028	2.523	2.378	2.245	2.048	1.828	2.452	1.907	2.132	2.196	2.322
thr60	1.676	1.547	1.511	1.457	1.366	1.313	1.496	1.287	1.387	1.448	1.52
thr70	1.284	1.218	1.198	1.212	1.134	1.154	1.187	1.102	1.127	1.17	1.175
thr80	1.117	1.135	1.078	1.101	1.057	1.033	1.054	1.037	1.057	1.067	1.087

Table 5: AUC scores for leave-one out cross-validation on the whole genome

	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN2	KNN4	KNN6	KNN8	KNN10
KNN1	0.67	0.629	0.638	0.636	0.628	0.581	0.572	0.624	0.646	0.638	0.635
KNN3	0.792	0.714	0.735	0.747	0.742	0.704	0.732	0.749	0.745	0.729	0.732
KNN5	0.818	0.753	0.768	0.78	0.785	0.762	0.775	0.775	0.77	0.761	0.755
KNN7	0.829	0.772	0.784	0.793	0.787	0.776	0.783	0.792	0.774	0.774	0.772
KNN9	0.838	0.777	0.792	0.79	0.792	0.772	0.783	0.794	0.786	0.777	0.774
KNN_sy1	0.74	0.677	0.692	0.71	0.71	0.662	0.657	0.716	0.711	0.7	0.696
KNN_sy3	0.84	0.774	0.789	0.797	0.798	0.772	0.788	0.799	0.794	0.782	0.782
KNN_sy5	0.848	0.792	0.807	0.813	0.807	0.775	0.79	0.799	0.796	0.798	0.789
KNN_sy7	0.852	0.796	0.812	0.817	0.802	0.783	0.786	0.805	0.803	0.793	0.787
KNN_sy9	0.857	0.801	0.814	0.82	0.811	0.786	0.797	0.805	0.797	0.799	0.792
thr40	0.848	0.794	0.803	0.817	0.808	0.772	0.779	0.799	0.8	0.793	0.783
thr50	0.774	0.73	0.743	0.747	0.754	0.724	0.723	0.735	0.728	0.736	0.737
thr60	0.655	0.629	0.636	0.646	0.627	0.617	0.614	0.639	0.636	0.633	0.635
thr70	0.58	0.58	0.562	0.576	0.554	0.553	0.561	0.559	0.567	0.572	0.574
thr80	0.536	0.534	0.547	0.533	0.525	0.526	0.515	0.523	0.528	0.529	0.528

F

Table 6: null-prediction counts for leave-one out cross-validation on the whole genome

	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN2	KNN4	KNN6	KNN8	KNN10
KNN1	569	656	777	958	1157	1335	1384	1158	935	808	733
KNN3	61	129	254	417	615	890	813	572	426	319	257
KNN5	40	95	215	367	536	733	630	465	351	267	213
KNN7	32	82	197	342	507	694	592	437	327	248	196
KNN9	32	80	193	337	498	677	582	427	318	242	192
KNN_sy1	232	283	401	576	767	1090	1068	710	547	438	379
KNN_sy3	30	77	189	332	492	669	573	421	313	238	188
KNN_sy5	30	77	189	332	492	669	573	421	313	238	188
KNN_sy7	30	77	189	332	492	669	573	421	313	238	188
KNN_sy9	30	77	189	332	492	669	573	421	313	238	188
thr40	44	90	201	343	500	676	583	431	327	251	202
thr50	278	358	462	588	730	876	809	669	569	506	460
thr60	719	874	944	1031	1140	1240	1233	1108	1037	981	946
thr70	1042	1233	1280	1329	1394	1445	1458	1400	1348	1301	1278
thr80	1233	1440	1463	1495	1536	1551	1555	1536	1505	1483	1467

Table 7: Perfect ranking counts for leave-one out cross-validation on the whole genome

	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN2	KNN4	KNN6	KNN8	KNN10
KNN1	145	154	157	156	155	149	141	142	144	147	151
KNN3	186	203	211	207	205	200	184	184	191	197	202
KNN5	188	210	218	214	206	200	180	182	188	198	201
KNN7	193	215	222	223	213	202	186	186	192	202	207
KNN9	191	216	222	218	215	206	184	184	191	201	207
KNN_sy1	183	183	187	187	184	178	167	169	171	176	181
KNN_sy3	201	210	219	216	212	207	189	190	196	205	211
KNN_sy5	207	221	227	227	221	215	193	194	200	210	215
KNN_sy7	209	229	233	231	226	216	197	198	204	213	219
KNN_sy9	209	229	235	229	225	216	199	198	204	210	219
thr40	190	195	202	199	196	187	165	167	173	179	190
thr50	186	205	211	206	205	196	180	178	184	190	201
thr60	123	138	141	144	139	134	110	114	125	132	136
thr70	76	87	89	91	85	83	55	64	74	79	82
thr80	37	48	50	52	44	42	14	27	36	39	42

H

Table 8: Average enrichment scores for leave-one out cross-validation on the whole genome

	PPI	thr_50	thr_60	thr_70	thr_80	thr_90	KNN2	KNN4	KNN6	KNN8	KNN10
KNN1	1.526	1.345	1.422	1.366	1.329	1.223	1.186	1.377	1.418	1.421	1.374
KNN3	2.383	1.763	1.905	1.971	1.93	1.767	1.846	1.99	1.95	1.865	1.834
KNN5	2.742	2.029	2.131	2.283	2.205	2.026	2.236	2.245	2.174	2.094	2.029
KNN7	2.94	2.213	2.33	2.392	2.348	2.191	2.336	2.348	2.293	2.229	2.124
KNN9	3.107	2.242	2.394	2.425	2.424	2.215	2.278	2.286	2.32	2.184	2.178
KNN_sy1	1.911	1.548	1.638	1.705	1.729	1.462	1.493	1.83	1.714	1.671	1.666
KNN_sy3	3.145	2.219	2.435	2.574	2.459	2.253	2.437	2.473	2.439	2.341	2.273
KNN_sy5	3.277	2.389	2.564	2.752	2.579	2.245	2.398	2.516	2.502	2.402	2.309
KNN_sy7	3.432	2.458	2.587	2.756	2.61	2.326	2.346	2.581	2.502	2.475	2.383
KNN_sy9	3.494	2.54	2.709	2.653	2.5	2.256	2.399	2.499	2.508	2.46	2.43
thr40	3.295	2.408	2.515	2.629	2.579	2.28	2.366	2.532	2.503	2.368	2.311
thr50	2.241	1.857	1.922	1.978	1.931	1.832	1.848	1.891	1.957	1.908	1.854
thr60	1.502	1.363	1.369	1.416	1.357	1.302	1.333	1.416	1.388	1.395	1.381
thr70	1.228	1.17	1.158	1.168	1.146	1.121	1.091	1.149	1.155	1.13	1.187
thr80	1.08	1.063	1.055	1.08	1.04	1.038	1.043	1.096	1.063	1.089	1.101

Table 9: AUC scores for parameter tests on artificial linkage interval

	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$
$\gamma = 0.4$,$\eta = 0.3$	0.861	0.856	0.861	0.866	0.861
$\gamma = 0.4$,$\eta = 0.5$	0.86	0.863	0.86	0.859	0.86
$\gamma = 0.4$,$\eta = 0.7$	0.862	0.865	0.864	0.864	0.862
$\gamma = 0.7$,$\eta = 0.3$	0.864	0.864	0.862	0.862	0.865
$\gamma = 0.7$,$\eta = 0.5$	0.861	0.867	0.864	0.862	0.863
$\gamma = 0.7$,$\eta = 0.7$	0.863	0.86	0.867	0.869	0.866

Table 10: AUC scores for parameter tests on the whole genome

	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$
$\gamma = 0.4$,$\eta = 0.3$	0.755	0.756	0.757	0.757	0.757
$\gamma = 0.4$,$\eta = 0.5$	0.755	0.756	0.757	0.757	0.757
$\gamma = 0.4$,$\eta = 0.7$	0.755	0.756	0.757	0.757	0.757
$\gamma = 0.7$,$\eta = 0.3$	0.757	0.758	0.758	0.758	0.758
$\gamma = 0.7$,$\eta = 0.5$	0.757	0.758	0.758	0.758	0.758
$\gamma = 0.7$,$\eta = 0.7$	0.757	0.758	0.758	0.758	0.758