Enhanced network GeneRank

Final Report for CS39440 Major Project

Author: Daniel Fryer (daf16@aber.ac.uk)

Supervisor: Dr. Chuan Lu (cul@aber.ac.uk)

4th May 2016

Version 1.6 (Final)

This report is submitted as partial fulfilment of a BSc degree in Artificial Intelligence And Robotics (GH76)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Acknowledgements

I am grateful to my supervisor Chuan Lu for all help and support throughout this project.

I’d like to thank Lee Harris for insight into the biology involved within GeneRank

Abstract

Microarray experimentation is done to determine the expression change between a reference and a “treated” sample where the reference will be based on a normal state and the treated could be from genetic changes or drug treatments. These experiments give a lot of statistical data based on the results. This can be combined with prior knowledge of the genes biological function to give more useful information about the results. GeneRank along with other ranking algorithms attempt to automate and improve on this process by creating a prioritised ranking list of genes based on the expression change in the experiment as well as network information of the genes.

GeneRank successfully improves upon ranking solely on expression change by integrating network data into the algorithm. The type of network information, as well as the amount of information directly affect how much of an improvement is made. Although clear improvements on ranking can be shown GeneRank is far from the best or most efficient algorithm for this and has been surpassed by modern machine learning algorithms. GeneRank suffers most on large scale experiments containing thousands of genes due to the structure of the algorithm.

For its time GeneRank was a clever and innovative idea which clearly benefited biological research by producing better ranking measures. However, as time has gone on others have taken the idea and improved upon it while avoiding some of the short falls that GeneRank suffers from.

Contents

Contents

1. Introduction

1.1 Human Genome Project (HGP)

The HGP was first publically promoted by Renato Dulbecco in an article published in 1984 who argued that sequencing the human genome would greatly enhance the understanding of cancer [1]. This historic goal was achieved in 2001 by a collaborated effort of 20 different groups. It was at the time, the largest genome to be extensively sequenced [2]. This achievement paved the way for a new era of biology with continuous efforts to sequence new genomes. The effects can be seen clearly in medicine where new approaches are being developed which rely on knowledge and understanding of genetic variation [3]. In the fourteen years since the sequencing of the human genome, vast amounts of data covering various genomes has been collected and given rise to large, open-access databases [4]. A direct result of this is the increasing use of Bioinformatics as a tool to manage and exploit this new found biological information. Bioinformatics expanded due to the human genome project, however, while there has been clear progress, bioinformatics is yet to be fully utilised [5].

1.2 Model Organisms

The term model organism is historically given to species such as the mouse *Mus musculus*, because they are small and have short generation timeframes [6]. Such organisms are usually fairly simple and allow for simulations of biological processes found in other organisms, in the case of medical science these processes are thought to be shared by humans [7]. The genomes of these organisms have been sequenced, meaning that there is vast amounts of knowledge about the genome and biological processes within it. The mouse organism has a database containing such information [8]. The mouse is widely used as a model organism as it has the common characteristics of a model organism, and as a mammal, has stronger similarities to the human genome. The mouse genome has roughly 22,000 genes, a number close to the roughly 20,000-25,000 genes in humans. The exact number of genes in the human genome is unknown and widely debated [9].

Furthermore the mouse is commonly used as a model organism as it is more ethical to conduct research on mice than it is to do the same research on species closer to humans, such as monkeys. The combination of the above characteristics of mice is the reason that the mouse is widely used as a model organism for medical and biological science.

Yeast *Saccharomyces cerevisiae* is another model organism, while not as closely related to humans as mice, yeast still shares many genes with humans [10]. David Botstein *et al*. discovered that for 31% of the potential protein encoding genes within the yeast genome there is “a statistically robust homolog among the mammalian protein sequences” [10]. The yeast genome has been sequenced, and has roughly 6000 genes. Yeast has a very rapid reproductive cycle and with a smaller genome has been used widely for cell biology research [11]. The term model organism has evolved to include organisms which share unique traits or characteristics which are deemed worthy of research [6].

The two other commonly used model organisms are the common fruit fly *D. melanogaster* and the roundworm *C. elegans* [6] [12] [13]. These model organisms share the common traits of model organisms and were amongst the first organisms to have had their genomes sequenced [13].

1.3 Microarray Experiments

Microarray experiments allow for profiling of gene expression and can in medical science give vital information of the specific features of a disease [14]. Today, expression level data is commonly used to explore causes and treatments for cancer. For this purpose, as well as for all other uses of expression data there is a dire need to process the vast quantities of data [14] [15].

A large use of microarray experiments, is to find the fold-change of gene expression between two experiments with one being a reference state and one the test or treated state. The data which comes out however, needs to be processed. There is a real risk that due to the scale of data coming from microarray experiments that based on bad statistical analysis wrong conclusions could be drawn from the data [16].

In many experiments, expression data is required from large numbers of tests in order to give validity to results. Over the course of several different treatments the tests performed can reach into the hundreds. The resulting expression data contains a lot of noise, as for instance, the same experiment performed on the same tissue sample could give different expression level results at different times. Different expression levels on the same tissue sample has also been observed when using different microarray technologies [16]. The preparation of tissue samples and the use of different dyes can also affect results.

As a result of noise, the significance of fold-changes in expression levels is also not always clear [16]. The change in expression values can however coincide with observed characteristics of a disease as found by S Hanash and C Creighton [14]. Although expression changes discovered by microarray experiments can lead to valuable information backed up by statistical significance, combining the expression results with prior knowledge of network data or observed characteristics leads to more founded results.

Therefore, results from microarray experiments may be sufficient alone in certain areas however, for many functions prior knowledge is needed as well as expression results are not perfect. This is due to the noise and uncertainty that is inevitable in results from microarray experiments. The use of prior knowledge increases confidence that the results are being interpreted properly with statistical and biological significance being accounted for.

The issues from large scale microarray experiments and the data produced is well known and there are various methods used to make sense of and interpret large scale data. For instance there are methods used to deal with missing values as well as various bioinformatics approaches which manipulate large scale data to reveal patterns and biological processes [17] [18]. There are also various databases containing expression data for organisms, such as the mouse [19]. Bioinformatics approaches to extract the valuable information that comes from microarray experiments are continually being developed and improved.

1.4 PageRank

The success of the Google search engine was largely determined by the success of PageRank [20], the algorithm used to determine the order in which webpages are displayed after a search. This order is based on a combination of key words used within the search as well as connections between websites and web pages. In this instance connection between pages was defined by hyperlinks. For every web page there is a node in the graph with an out degree and in degree defining how many pages the page links to and how many pages link to that page [21]. If purely the terms searched for were used, the volume of results would be staggering. PageRank is used to prioritise the results of the given search. The more hyperlinks linking from other web pages to a web page containing the key search terms, the higher the ranking given.

1.5 GeneRank

The purpose of GeneRank as well as all subsequent ranking algorithms is to prioritise genes for further analysis based on biological experiments [22]. This is one of many ways of getting valuable information about biological process from the large scale expression data which comes from microarray experiments. An example use would be to determine a prioritised list of genes within the human genome, which have an expression change between a reference and a treated experiment. This could give vital information on the processes which occur when a treatment is used, for example, to treat cancer. With the example of cancer, there is a high number of candidate genes which are thought to alter cellular processes [23]. The purpose of algorithms such as GeneRank is to give a faster and better prioritisation of such candidate genes which require more in depth research. This is achieved through automation of the process.

The idea was put forward by Morrison *et al.* to adapt the approach used for PageRank, in order to rank genes based on the connections of genes within the genome [22]. GeneRank combines the expression level of the genes with a connectivity matrix. When ranking is based purely on expression data for a given experiment the ranking will not give the full picture. Some genes will be neither up-regulated nor down-regulated however their transcription factors could be activated. As a result some genes will be activated as they are controlled by these transcription factors. These genes, which affect the up-regulation and down-regulation of other genes, are clearly important and in some respects should be ranked more highly than those which only change expression level. These genes are accounted for by GeneRank but not by ranking based only on expression levels [22]. This is why the combination of prior knowledge in the form of connection data with expression data is such a powerful concept which improves gene ranking and continues to be used and developed by subsequent algorithms.

The connectivity matrices are built from connectivity data which can come from a range of sources. For the algorithm, If two genes are connected then the associated element within the matrix is set to 1, if the two genes are not connected then this is set to 0. The algorithm has a weighting value d, which determines the influence connection data has on the final ranking list. If d is set to 0 then the algorithm bases the ranking purely on expression data, If d is set to 1 then the ranking is based purely on connection data, and if d = 0 these two elements are given equal weighting.

It is important to note that for the GeneRank algorithm to work effectively the absolute expression value must be used, therefore, if the expression change is negative as a result of down-regulation it must be converted to a positive number. As a result an expression value of -1 would be changed to +1. This allows the algorithm to rank based on total change as without this addition down-regulated genes would always be ranked lower than up-regulated genes.

1.6 Limitations of GeneRank

GeneRank had limitations in its approach in using this connection data. The main approach suggested and implemented with testing and evaluation was to use GO (Gene Ontology) annotations as a source of connection data. This approach has been used previously by others with success [14] [24]. Two genes are connected if they both share the same GO annotation. As mentioned by Eugene Demidenko [25], this can become an unrealistic approach to use in practice as large numbers of genes require massive connectivity matrices. As an example, with the yeast genome, containing 6,000 genes, 6million connections can be found when using GO annotation. This leads to a graph of around 286KB in size, unreadable by programs designed to show large scale graphs visually. The creation of such a graph takes several minutes as every genes GO annotations is matched against ever other genes GO annotations.

The scaling problems for larger genomes continue into the algorithm itself, which updates the ranking of every gene n times, with n being the number of genes. For the first iteration all genes are updated against the initial ranking of the first gene. For the second iteration all genes are updated against the ranking of the second gene. This means that for iteration n all genes are updated according to the rank of the next gene at iteration n-1. This means that the algorithm itself can take several minutes when 6,000 genes are used. An additional side-effect of this approach is that the order in which the genes are entered into the algorithm will affect the resulting prioritised ranking list. This idea is further shown in demonstration below where the mouse genome containing 22,000 genes was used. GeneRank is better suited to smaller genomes, as the larger the genome the longer the algorithm takes exponentially and the worse the resulting ranking down to order of genes.

The difficulty in using GeneRank for each new experiment is also discussed by Eugene Demidenko, as for each experiment there is new expression data and so the algorithm has to run again, which could take a long time over multiple experiments [25].

Morrison *et al.* suggest that when d is set to 0.5 it produces the best results and so should be used as the standard weighting for the algorithm [22]. Eugene Demidenko also however, states that having the threshold parameter d set to 0.5 is unjustified [25]. Further evaluation within this study shows that in fact a value of 0.95 or 1 for the weighting d gives the best results on different data. This is similar to results that Morrison *et al* discovered using synthetic data where d values of 0.75 to 0.85 performed best, they also note that d = 0.85 is reportedly used by Google for PageRank [22]. It is curious then that for general use they determine that a d value of 0.5 is suited best, stating that it improved ranking while no deterioration was observed. A d value of 0.5 is however, provably non-optimal and perhaps not best suited for most uses of the algorithm.

1.7 Evaluation and proof within GeneRank

There are two evaluation methods used within the GeneRank paper, a synthetic network and Correlation Coefficient Networks as shown in the data set [26]. Using the synthetic networks they prove that ranking with a high weighting on connection data is significantly better than ranking based solely on expression levels. This is done by comparing two sets of genes, where it is known that one set, setA should be ranked higher than the other, setB. When compared during ranking via the GeneRank algorithm a receiver operating characteristic (ROC) score result is used [27] [28]. The ROC value given is 0.5 if the two sets are mixed and 1 if perfectly separated. Therefore in this experiment the higher the score, the better the ranking as setA contains genes known to be higher than those in the setB.

The “Correlation Coefficient Networks” evaluation is used to support the idea that GeneRank has given a better ranking measure when connectivity data is included and highly weighted. This in itself is an achievement however it does not suggest that genes within each set, for instance in setA which is known to be ranked higher than genes in setB, are ranked well. It could be that although genes in setA are generally ranked higher than those in setB as they should, genes in setA may not have a good ranking within the set.

It is clear that GeneRank can run, and perform well on genomes of up to 6,000 genes such as the yeast genome. The algorithm used provably gives a better prioritised ranking list of genes than the common approach at the time of using only expression values for the same ranking. This was perhaps one of the first algorithms to achieve this end and lead to continued improvements and alternative ranking algorithms [25] [29]. A MatLab implementation of the algorithm remains free and open access allowing others to use it for research or improve upon it. Since the success of GeneRank alternatives with use similar approaches out perform this, however, this is to be expected as time passes. GeneRank remains a good option for small scale genomes and is easy to implement and use.

1.8 Enhanced Network GeneRank

The benefits of having a better prioritised ranking list of genes from experiments as discussed above are of huge value to biological and medical research. As such there are continuous efforts to improve on the idea of GeneRank. The purpose of Enhanced network Generank is to evaluate the GeneRank algorithm against modern approaches and to implement and suggest further improvements. This is done by following the basis of experimentation done within the GeneRank paper and combining that with modern evaluation techniques with proven alternative approaches to ranking genes [29].

The GeneRank algorithm is re-written in Python using Anaconda 3.5 [30] as the original is written in MatLab which requires a user license. The algorithm is embedded in a program designed for ease of use. This is done so that the algorithm is more accessible, and comes with several different versions. The aim is to provide a user friendly library containing the GeneRank algorithm which will produce a prioritised ranking list of genes as well as a graph based network file which is viewable using Cytoscape [31]. This allows for visualisation of the highly prioritised genes within the network structure of the genome which has been identified and used for the ranking.

1.9 Modern Evaluation Techniques

With the current evaluation methods used, it cannot be determined whether individual gene rankings can be considered good, but more so that the overall ranking of all genes is improved upon. Therefore instead of using the same evaluation methods, Enhanced Network GeneRank focuses on the use of the methods used by Daniela Nitsch *et al.* [29]. This modern approach is used to evaluate several machine learning approaches to the gene rank problem in order to determine which one produces the best rankings. This allows the GeneRank algorithm to be directly compared to other approaches in order to determine whether or not GeneRank produces rankings good enough to warrant its use in the future.

For this evaluation 40 subsets of genes within the mouse genome are used. The mouse is a model organism as described above, and is used in biology frequently including use as a model for development of drugs developed for human use [6]. As a result, the mouse genome is a good test case for GeneRank and all alternative ranking measures. If it is possible to effectively rank genes within the mouse genome for a given experiment or treatment then this could lead to more effective and targeted research. The mouse genome consists of around 22,000 genes, for the evaluation of various ranking methods, 40 subsets of 100 genes were used [29]. This favours algorithms such as GeneRank which do not perform so well on large scale genomes however remains a fair evaluation as this type of experiment with 100 to 1,000 genes are target at one time due to prior knowledge of the genome.

Each subset contains expression data for 100 genes based on an experiment where one gene was knocked out, called a KO Gene. Various biological research projects use KO mice [32] [33]. This is an area in which ranking algorithms could aid in biological research. The 100 genes used for each evaluation are the “nearest neighbours” to the KO gene [29]. The expression values for all genes within the subset is therefore based on this KO gene. As a result the ranking of the KO gene should ideally be within the top 10, as the KO gene caused the other genes changes in expression levels.

The number of times the KO gene is in the top 10 ranked genes, over all 40 experiments gives an indication of the algorithms performance. As a second measure the same count is done for the number within the top 20 ranked genes and as a third measure a ROC score is taken for each subset. The ROC score is calculated with every KO genes expected ranking score set to 1 with all others set to 0 for the GeneRank algorithm, all ranked genes should have a score between 0 and

1, the higher the score, the higher the ranking. The ROC score compares the expected outcome against the real ranking score produced for each gene in the subset. For further analysis of the GeneRank algorithm, these three evaluation scores, top 10 count, top 20 count and roc scores, were measured across all values of d. Where 0 <= d <= 1.0 in increments of 0.05.

The average ranking across the 40 files is another indication of how the algorithm performs. The methods used by Daniela Nitsch *et al.* largely achieved an average ranking of below 10 with the best algorithm having an average rankinf of 8 [29]. In comparison the best average ranking the GeneRank algorithm produced, across all values of d and several different networks, was 25. This is however, much better than the average ranking of 42 which is produced when only expression data is used as a ranking measure.

1.10 Observations from Evaluation

Experimentation on using different network data within GeneRank shows that although different networks have some effect on ranking, the difference on average ranking is minimal. However, the network used can have a larger impact on whether the gene is ranked in the top 10 genes or not. Overall within GeneRank it is clear that for ever increasing values of d, the weighting for network information, the better the average ranking and likelihood of a gene being in the top 10 increases dramatically. This continues as a trend until d is 0.95 or 1, with some networks giving better scores for a d value of 0.95 while others give better rankings when d is 1 and so the expression data isn't considered at all. When networks are combined, GeneRank gives a better ranking. Overall, there is clear evidence that GeneRank does improve upon the system of ranking purely on expression data. This shows, that although not as good as modern approaches, GeneRank performs the function it was designed to do, and does so consistently across different networks.

1.11 Connection data

The connection data for a genome can come from a variety of sources, the main focus of the GeneRank paper is on GO annotations [22]. Noting that GO annotations have been successfully used in previous papers [14] [24]. Other methods such as Protein-protein interactions have been also used previously [34] [35]. GO annotation ID's for genomes can be found on the GO consortium website [36]. From here files containing the genes along with their GO ID's can be found for many different organisms, including many model organisms. Connection by GO annotations, if two genes within a genome share a GO term then the two genes are connected. On a genome such a Yeast with 6000+ genes this gives around 6 million connections which as discussed above, can be unrealistic to create and use in real practice due to size.

The use of GO Ontology data is a more proven concept and is used for analysis of the algorithm within the Synthetic data section of the GeneRank paper [22]. GO Ontology is split into three sections defined by the GO Ontology Consortium [36]. The three Ontologies are: Cellular Component, Biological Process and Molecular function. Each Ontology has GO ID's associated, genes which share the same GO ID's for example Biological Process, will share the Biological Process defined by that

GO ID. For this network structure genes sharing a GO ID should be connected within the network. This allows for three unique networks to be created as well as the possibility of combined networks where network information from all three, or any combination of the three can be used. Again, downloads for these networks can be found on the GO Consortium website.

The use of multiple networks based on different connection data allows for more in-depth testing of the algorithm and could lead to better results, if for instance one network is proven to be better than others. During the analysis of the 40 subsets of data used in this study, when all three ontologies were combined the graph size ranged from 24-63KB which is a manageable size.

Another source of connection data is Protein-Protein interactions whereby two genes are connected by protein interactions [37]. The data showing the connections by protein-protein interactions comes from the String database [38]. The connection can be based on a number of sources and measures, such as experiments, databases, neighbourhood, textmining, co-expression, co-occurance and gene fusion. Each connection found by these sources and measures will have an associated confidence score. The higher this score is, the more certain it is that a connection between the two genes exists.

Protein-Protein interaction based networks can be used as standalone networks or combined with the above GO Ontology networks. There are also other methods for connection genes based on prior knowledge, however, these were not used for evaluation of the GeneRank algorithm and usually require in-depth study or the use of databanks [39] [40]. The purpose of all gene network and connection based projects is to further understand the interactions between genes and the biological processes involved. For this end, the more connection data available for a given genome, the more prior knowledge available. It is this prior knowledge which modern ranking algorithms use to create a better prioritised ranking list for a given experiment.

1.12 Degree of Certainty

It is important to note that for both gene expression data and gene connectivity data there is an associated degree of certainty. As discussed above, noise within gene expression data can reduce certainty about results. This however, remains true for connection data, the connections found between genes, using any of the methods defined above will have a degree of certainty. Some connections between genes are more certain than others. For this study the certainty of connections is not ensured, and so may not be fully reliable. Due to the size of the networks that are used as well as the combination of network data and expression data, this should not affect the end prioritised ranking list in a meaningful way. This is however an area which could be expanded on and evaluated in itself. Where networks with higher levels of certainty could potentially change the end rankings.

1.13 Alternatives Approaches

Several subsequent studies have shown ways to improve upon GeneRank itself, as well as using the same principle idea of combining expression data with connection data with other machine learning approaches [25] [29]. There are different ways in which it is possible to incorporate both data types in order to produce a prioritised ranking list. The algorithms are also affected by the quality of connection data used as well as the expression data. Gene Ranking and especially network ranking as a method has been used by various other projects [41] [42]. Many of these approaches focuses on reducing the false discovery of connections, for this reason Eugene Demidenko states that “gene connectivity can be adequately expressed in terms of the gene pairwise squared correlation matrix” [25] [43] [44]. Microarray data is usually correlated to a phenotype, however as connectivity can come from other sources this is not always required. Phenotypes can be examined after the ranking results.

It is clear that since the implementation of this style of ranking introduced by Morrison *et al.* further study and improvements have proven how useful it is for biological research [22]. There are however, other methods that achieve the same end goal, of focusing research on prioritised elements within a genome. This can be achieved by the study of networks, sub-networks and clustering. By looking at the network structure and clusters within that structure, it is possible to identify important genes as well as important sub-networks [45]. Network clusters can be created for a given experiment or for the reference state of the genome. Both can give great insight into the biological process [40].

2.0 Methods

2.1 Expression Data

The expression data came from 40 publically available datasets from Affymetrix chips on mice with and without KO genes as used by Daniela Nitsch *et al.* [29]**.** The data from these files was pre-processed by Daniela Nitsch et al. and several statistical measures were given for the expression change data. For all experiments done within this study, the “logFC” value was used. Table 1 shows the data sets used for evaluation. The 40 KO gene files used differs by one gene from the 40 used by Daniela Nitsch as the Prkag3, AMPK G3 gene is replaced by Tcf7 due to availability to data [29].

Table 1. Showing the 40 KO genes used.

No.	Gene Name	No.	Gene Name
1	Abca1	21	Mbnl1
2	Btk	22	Mstr, Ron
3	Cav1	23	MyD88
4	Cav3	24	Nos3, eNos
5	Cftr	25	Phgdh
6	Clcn1	26	Pmp22
7	Cnr1	27	PPARα
8	Emd	28	Pthlh, Pthrp
9	Epas1, Hif-2	29	Rab3a
10	Esrra	30	RasGrf1
11	Gap43	31	Rbm15
12	Gnmt	32	Runx
13	Hdac1	33	Scd1
14	Hdac2	34	Slc26a4
15	Hsf4	35	Srf
16	Hspa1A, Hsp70.1	36	Tcf7
17	IL6	37	Tgm2
18	Lhx1, Lim1	38	Zc3h12a
19	Lhx8	39	Zfp36, tpp
20	Lmna	40	Zfx

2.2 Network Creation

The network creation is somewhat dependant on the structure of the network data. For all GO data, for both GO annotations and the three GO networks the same method is used. Each gene has an element in a list containing all its associated GO ID's. This list is in the form of [GeneName, GO ID's, GeneName, Go ID's,…] and contains all genes and their GO ID's in the order they appear in the file read. Using this structure, each gene has its GO ID's compared to that of every over gene, if one of the GO ID's matches then the genes are connected and a link is added in the graph.

The protein-protein interaction network was created using the String 10.0 dataset [38]. This was done using the "Multiple Proteins" section, with the ~100 genes from each of the 40 KO gene files put in the list of names section. The *Mus Musculus* organism was selected as these files all contain genes from the mouse organism. When the final graph was available, the table produced was exported. For the use of protein-protein interaction networks within this study, only experiments was used as a source for network connections. This however, could be at any confidence value.

For the protein-protein network, the data file which is read in has two columns, each column has a list of genes. For each line the genes in column 1 and column 2 are connected in some way, it is possible therefore to add a link in the graph for these two genes very simply. Al network connections were created using the add_node function found in the networkx package [46].

An example network graph is shown below in figure1. This is the network graph for the file for the KO gene Abca1 using the protein-protein interactions data.

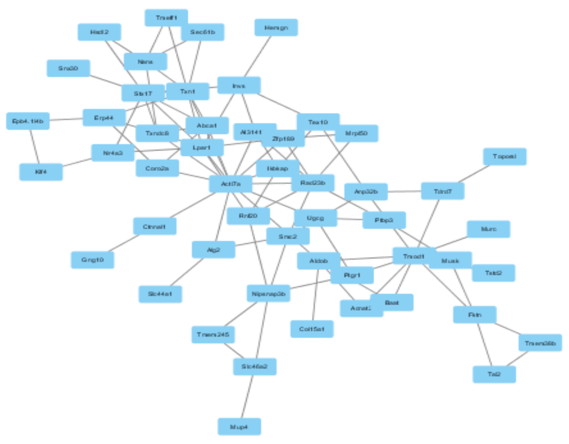


Figure 1
Showing the “PToP” Network for the Abca1 KO gene file. Each node is a gene in the file.

The graph is built using the GO Ontology network information use the probe ID as nodes in the graph. If a gene is duplicated however, the average expression change value is used. This value will replace the expression change value of the first probe for that gene and delete all data for the other probes that are for the same gene. The written file from the python file designed for use by biologist’s prints out the name of the gene alongside the probe name. This is done as some genes have multiple probes and although in this study we remove duplicate probes and use the average expression change, it may be desired for a gene to have multiple probes in the same experiment.

Using the probe ID for nodes in the graph allows for multiple probes for the same gene to be ranked at the same time. This will affect the rankings of all probes as adding more probes will add more elements into the GeneRank algorithm. The two probes for the same gene will have different rankings as long as the expression change value is different. This is not done for the protein-protein network due to the nature of the network information.

2.3 GeneRank Algorithm

The main aspect of the algorithm is a python implementation of the original GeneRank algorithm designed by Morrison *et al.* which is defined below. This ensures that the evaluation of the algorithm is based on the same algorithms put forward. The main aspect of change is the data put in, the network construction, evaluation technique and supporting code which allows for easy repeatable use of the algorithm. The algorithm is based on the following:

$$r_j^{[n]} = (1-d)\mathrm{ex}_j + d\sum_{i=1}^{N}\frac{w_{ij}r_i^{[n-1]}}{\deg_i}, \quad 1 \le j \le N. \qquad [22]$$

As suggested by Morrison *et al.* all genes are given an initial ranking $\mathbf{r}^{[0]} = \mathbf{ex}/||\mathbf{ex}||_1$, where $||\cdot||1$ denotes the vector 1-norm. Ex is the absolute value of the expression change for the gene. W_{ij} is based on connectivity whereby, if the genes i and j are connected then a value of 1 is given and if they are not connected a value of 0 is given [22]. In place of the connectivity matrix used Morrison *et al.* the connection data is stored in a graph created using the Networkx tool [46].

2.4 Python Implementation of GeneRank

The python implementation of GeneRank was developed using the list data structure which allows for the method to be used as a standalone method or as part of the script created.

```
geneRank(geneNameList, exprDataList, normExprDataList, G, d)
where G is the graph created using networkx and d is the weighting value. The three
lists are must be in the same gene order, where element [i] for each list refers to
information for the same gene.
sumOfConnectionList = []      // two lists for the other required values
rankingValueList = []
num3 = len(geneNameList)   // all the list have the same number of ellements
i = 0
j = 0
while (i < num3): // two while loops create r_j^[n]
    while (k < num3):
        if (i == 0):      // create the other two required lists on the first run
            sumOfConnectionList.append(0)
            rankingValueList.append(normExprDataList[j])
            // this gives each gene their initial ranking score
            elif (i != j):      // as long as i and j aren't the same gene
                if there is an edge between genes i and j in the graph
                    set hasEdge to 1
                else
                    set hasEdge to 0
                if gene i has no connections then
                temp_connection = (hasEdge * (rankingValueList[i-1])) / 1
                else
                temp_connection = (hasEdge * (rankingValueList[i-1])) /
                                                (G.degree(geneIDList[i]))
                sumOfConnectionList[j] = sumOfConnectionList[j] +
                temp_connection
                connectionValue = (1-d)*exprDataList[j]
                rankValue = connectionValue +
                d*(sumOfConnectionList[j])
                rankingValueList[j] = rankValue
        j = j + 1
    i = i + 1
    j = 0   // reset j so that every gene is compared again next iteration
return rankingValueList      // The final prioritised ranking list
```

With sumOfConnectionList being the equivalent of $\sum_{i=1}^{N} \frac{w_{ij} r_i^{[n-1]}}{\deg_i}$ for each gene.

Some genes within the KO gene files had multiple expression level results. In this case the average expression value was used.

2.5 Mothods Of Evaluation

The evaluation is made up of an ROC analysis as well as the average rank and top10 counts for each value of d over the KO gene files. For the ROC analysis each KO gene is given an expected score of 1 and all others are given an expected score of 0. This is build up in a list containing expected scores for all KO gene files. Another list of the true rankings is also created, these two lists are used for the roc analysis after being converted to being np arrays. The roc_auc_score is defined in sklearn.matrics [47].The average rankings and top10 counts are simply calculated based on the output data file created, this is shown in more depth in the experimental design section below.

3.0 Experimental Design

3.1 Original Design

The original design relied on the same datasets used by Morrison *et al.* In order to get an idea of whether or not the algorithm functioned according to the existing study [22]. This was going to be used as a basis for the design as well as an evaluation of the methods used by Morrison *et al.* The connection data for the yeast genome came from the Gene Ontology websites database and contained the GO annotation information for all genes within the yeast genome under the "Saccharomyces cerevisiae SGD Stanford University" download [48]. The data file downloaded was used as a basis to build the python implementation of the algorithm. The list data structure used had to be constructed correctly for the data being read for later parts of the algorithm. This allowed for a full design covering the reading of the data, building and adding connection data to the network graph, the GeneRank algorithm implementation and subsequently writing the output to a file.

As part of this design several other sections were required, such as creating the required lists for the geneRank algorithm. As well as the re-ordering of the list containing the genes and their ranking scores so that the first gene would be the highest ranked gene. The initial design did not contain a ROC analysis or average ranking scores. This prototype design created a random expression level for each gene, this prototype was to show that the algorithm would run and that some output could be produced. No real expression change data or resulting ranking list was available at this time. For this prototype the GeneRank algorithm is not structured for ease of use outside of the python file. Figure 2 shows the function structure of the original prototype design in the form of a flow diagram.

This design however had to change as it proved difficult to get the real expression data from the files provided by Morrison *et al.* All implementation provided was written in MatLab and although a MatLab to python converter, scipy.io.loadmat was used so that it was possible to access the data structures, it proved difficult to get out the raw expression change data required [49].

3.2 Adapted Design

The updated design was based on the method shown by Daniela Nitsch *et al.* [29]. The creation of the 40 KO gene files containing expression data and GO network information is described in the methods section of this paper. 40 publically available datasets from Affymetrix chips on mice with and without KO genes were used.

This new design had to make use of these new KO gene files which contained the logFC value as well as GO network ID's for each of the three GO networks. The programme structure designed for ease of use of the algorithm stayed much the same. However, in order to allow for the algorithm to be called for multiple files, and not just one example case some changes were made to the creation and structure of the lists used.

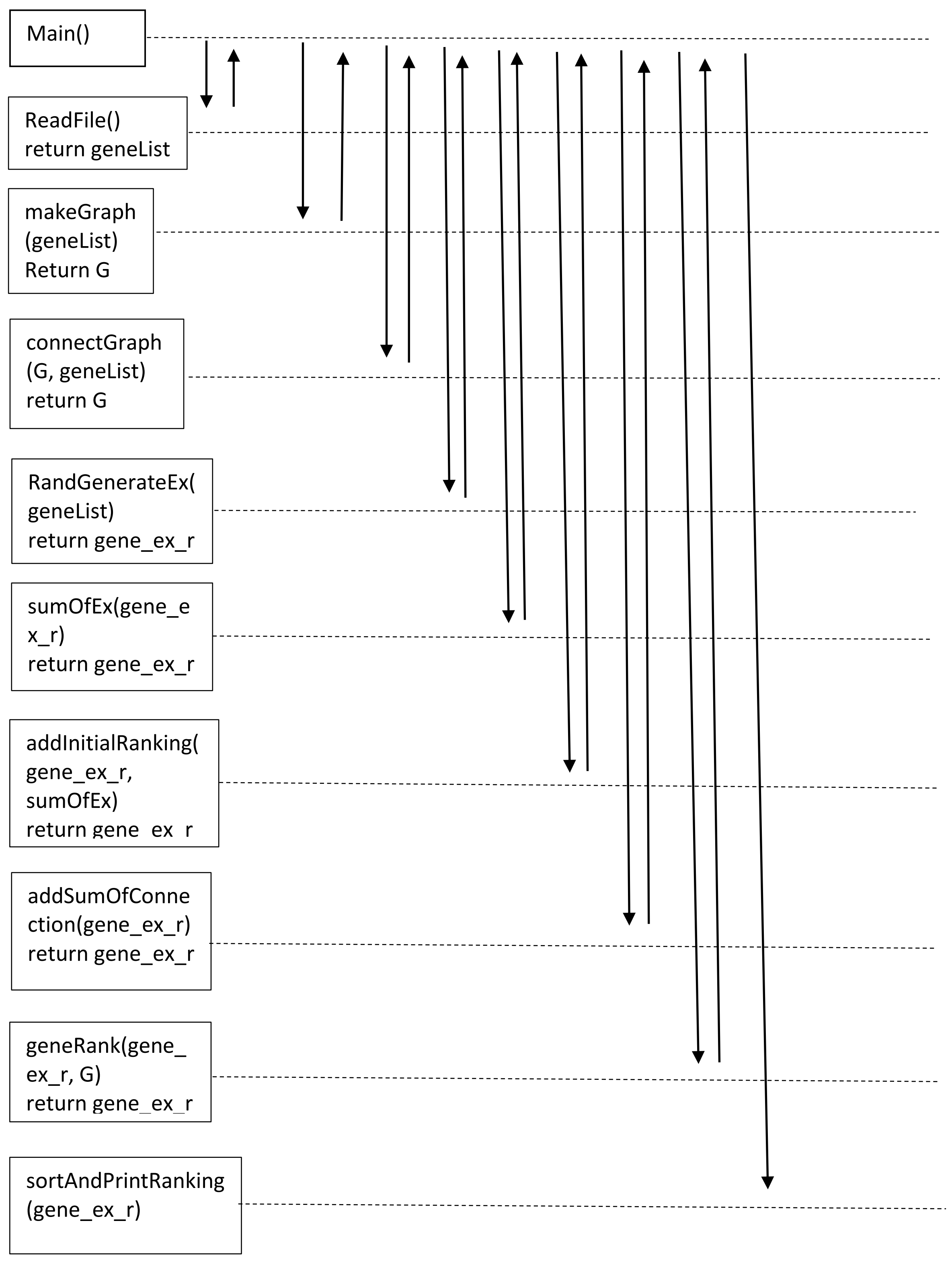

Main()
ReadFile() return geneList
makeGraph (geneList) Return G
connectGraph (G, geneList) return G
RandGenerateEx(geneList) return gene_ex_r
sumOfEx(gene_e x_r) return gene_ex_r
addInitialRanking(gene_ex_r, sumOfEx) return gene_ex_r
addSumOfConne ction(gene_ex_r) return gene_ex_r
geneRank(gene_ ex_r, G) return gene_ex_r
sortAndPrintRanking (gene_ex_r)

Figure 2. Showing a flow diagram for the python file. All methods are called from "main()" and return list elements. The final output is then printed by "sortAndPrintRanking(gene_ex_r)"

The GeneRank algorithm was modified such that it took in multiple lists. The lists are required to be in the same gene order, and be the same length, so each lists elements contain information for the genes in the order they appear in the file which is read in. This required the creation of new lists and such the design changed.

The modification of the original GeneRank method design allowed for it to be called multiple times within the programme. This was required for evaluation across all 40 KO gene files, as was the ROC score. Therefore the design also needed to include a way to ROC score also had to be calculated over the 40 files, for all values of d. All relevant data was to be appended to a file which would end up containing the ranking score for all values of d, across all 40 KO gene files. In order to speed up the process, one design allowed for file names to be entered into the program for the file to be read, and the file to write output to. These file names would have to be entered in turn, for all 40 KO gene files. However, another design had the KO gene file names and the desired output file names hardcoded in. This made debugging far quicker when the design was implemented. Figure 3 shows the amended function design structure in the form of a flow diagram. This is for the file which calculates the ranking for the KO gene for all values of d, for all 40 files. The hard-coded version simply had the file names to be read hard-coded.

In order to get some of the evaluation methods another script was designed to take the output data and calculate the average scores along with the top 10 and top 20 counts. Another file was used to calculate the roc score across all 40 KO gene files, using much of the same design for getting the raw data results. For this to be achieved a createValidityScores and a writeRocScore method was added to the design shown in figure 3 while the writeResultsToFile method was commented out. The creation and writing of the roc score results comes after the rankForAllD method which is left in. It is possible to change the file such that both the ranking and the roc score can be done on the same file.

3.3 Evaluation Method Design

Based on this new design, it was possible to use the same evaluation methods used by Daniela Nitsch *et al.* [29]. They use four different evaluation measures to evaluate the various machine learning algorithms implemented. These are the average rank, over the 40 KO gene files, the top10 count and the top20 count along with a roc score. By using the same data files which are used in this study it is possible to directly compare GeneRank to the modern methods described. The new data, along with expected results also means that it is possible to test the running of the algorithm itself, and to evaluate whether or not the algorithm is equivalent to ranking on expression value when d = 0. Over the 40 KO files any duplicate genes had their logFC values averaged, this average was used as the expression change value for the algorithm.

The design based on the 40 KO gene files allowed for a protein-protein interactions network to be implemented as data coming from String 10.0 allowed for the creation and use of files showing the various connections by protein-protein interactions. The addition of this network means that a more thorough investigation into GeneRank was possible.

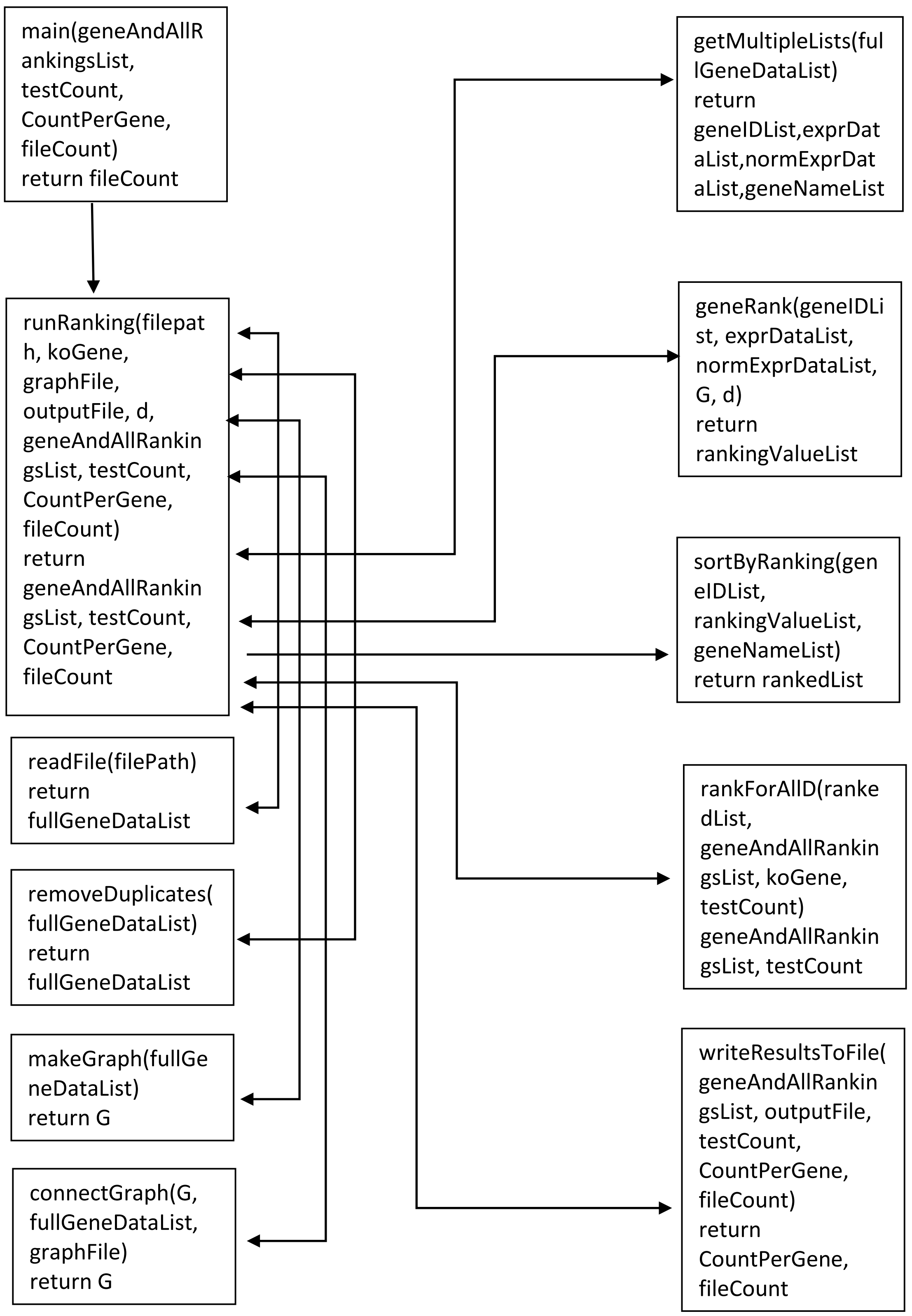

main(geneAndAllRankingsList, testCount, CountPerGene, fileCount) return fileCount
runRanking(filepath, koGene, graphFile, outputFile, d, geneAndAllRankingsList, testCount, CountPerGene, fileCount) return geneAndAllRankingsList, testCount, CountPerGene, fileCount
readFile(filePath) return fullGeneDataList
removeDuplicates(fullGeneDataList) return fullGeneDataList
makeGraph(fullGeneDataList) return G
connectGraph(G, fullGeneDataList, graphFile) return G
getMultipleLists(fullGeneDataList) return geneIDList,exprDataList,normExprDataList,geneNameList
geneRank(geneIDList, exprDataList, normExprDataList, G, d) return rankingValueList
sortByRanking(geneIDList, rankingValueList, geneNameList) return rankedList
rankForAllD(rankedList, geneAndAllRankingsList, koGene, testCount) geneAndAllRankingsList, testCount
writeResultsToFile(geneAndAllRankingsList, outputFile, testCount, CountPerGene, fileCount) return CountPerGene, fileCount

Figure 3. Showing a flow diagram for the python file which calculates the KO gene rank for all values of d for all 40 KO gene files. "main()" calls "runRanking" which calls the required methods in order. For the hard-coded version when main is called the file names have been entered already. Otherwise main asks for file names to be entered before moving onto calling runRanking.

It is possible from this to evaluate across 5 different network structures, for all values of d across the 40 KO gene files leading to a more robust evaluation than that used by Morrison *et al* [22]. Another python file was written to take the KO gene rankings for all 40 KO genes, across all values of d tested and create an average rank, top10 count and top20 count for each value of d. This design for this is shown in figure 4. This along with the roc score measure fives the full spread of evaluation measures.

For the protein-protein network the only changes were to the files for creating the rankings and roc scores for the 40 KO genes. The modifications for this are as follows. A readConnectionFile method was added, the readFile method was replaced with readExpressionFile. The getMultipleLists method was replaced by getNormalisedExprData and The removeDuplicates method was also no longer required. The design followed the same principle that runRanking was called by main and from runRanking all other methods were called. This was only done using a file with hard-coded filenames put into the main method however it would take very little modification to allow the user to enter these. The required code is in the file however, is commented out in favour of the hard-coded solution which during testing and evaluation required less time. All files which are hard-coded are only used for evaluation of the algorithm and are not intended for end-user use. They remain open source for further evaluation.

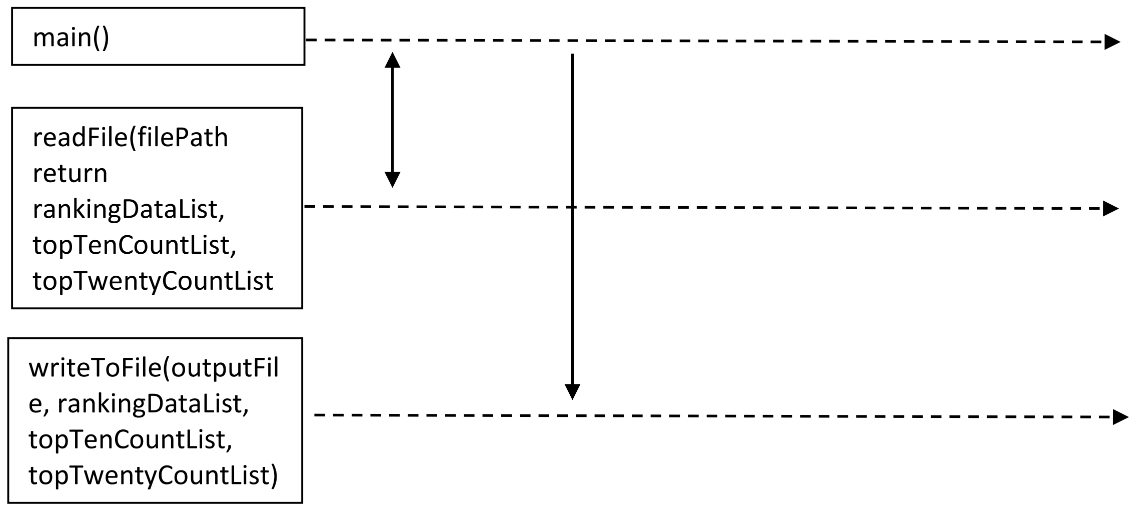


Figure 4. Showing the python file which calculates the average rank, top10 count and top20 count for each value of d, across the 40 KO gene files.

The end goal of GeneRank is to allow biologists access to a programme which will automate and thus speed up the process of priority ranking genes. The end result is proven to work however, especially without access to MatLab is difficult to use in practice. Therefore in this study a user friendly design was introduced to allow for easy use of the programme which could read and write files or take in raw data.

3.4 User Friendly Design

The original user friendly design included the evaluation method in the original design as the results were to be evaluated against the results found by Morrsion *et al*. Firgure 2 shows the original design which includes the reading of files, using a

GeneRank method and outputting the rankings in order. Also included is the creation of the graph file based on GO network information, graphs created will be written to an xml file.

As the evaluation design changed in order to use the 40 KO gene files, so too did the user design. The new design made use of the GeneRank method which could be called as a standalone, or as part of the programme. This simplified much of the design however as with the evaluation design it meant that when reading files several different lists had to be created. Figure 5 shows the final design which biologists would be able to use. It is written in Python such that anyone with programming knowledge can see the source code and adjust it as necessary. For instance, they may not want to write the xml graph file every time as this can take some time depending on size and may not be necessary in all cases.

This design focuses on GO network data however it would be simple to change this for other network types. As GO network information is arguably good enough and is widely available it is what this design uses. The design not only gives a full prioritised ranking list of the genes supplied but also writes the xml graph file for the network graph used. This could aid biologists by giving them access to the network structure used as well as the raw rankings. By combining this information it would be possible to see clusters of highly ranked genes in the network. The end goal is to provide as much useful data as possible to allow biologists to have the best idea of where they should focus future research.

4.0 Implementation

All implementation was done using the improved design based on the 40 KO gene files. The creation of the networks required as well as the gene expression data are described within the methods section. The end result was several different python files, one designed for ease of use for biologists which prints only the prioritised ranking list. One file to calculate the ranking of each KO gene for each value of d from 0 to 1 in intervals of 0.05.

This was based on the GO ID networks. A different file was created to deal with the protein-protein interactions network as this required reading in two files. However, it did not need as many separate features as the connections within these files were simpler to parse. These two files printed, one after another, lines containing the gene name followed by each rank position at each value of d.

A further file took the output of the files containing the rank position of the KO gene, for all 40 files, for all values of d. This then calculated the average rank of the 40 KO genes for all values of d as well as creating a top10 and top 20 count for each value of d. A final file was used to calculate the ROC score for each value of d over the 40 KO gene files.

The implementation was written fully in python using Anaconda 3.5 [30]. 10 python scripts were made and all are available. The appendices contains a link to a github repository containing these files.

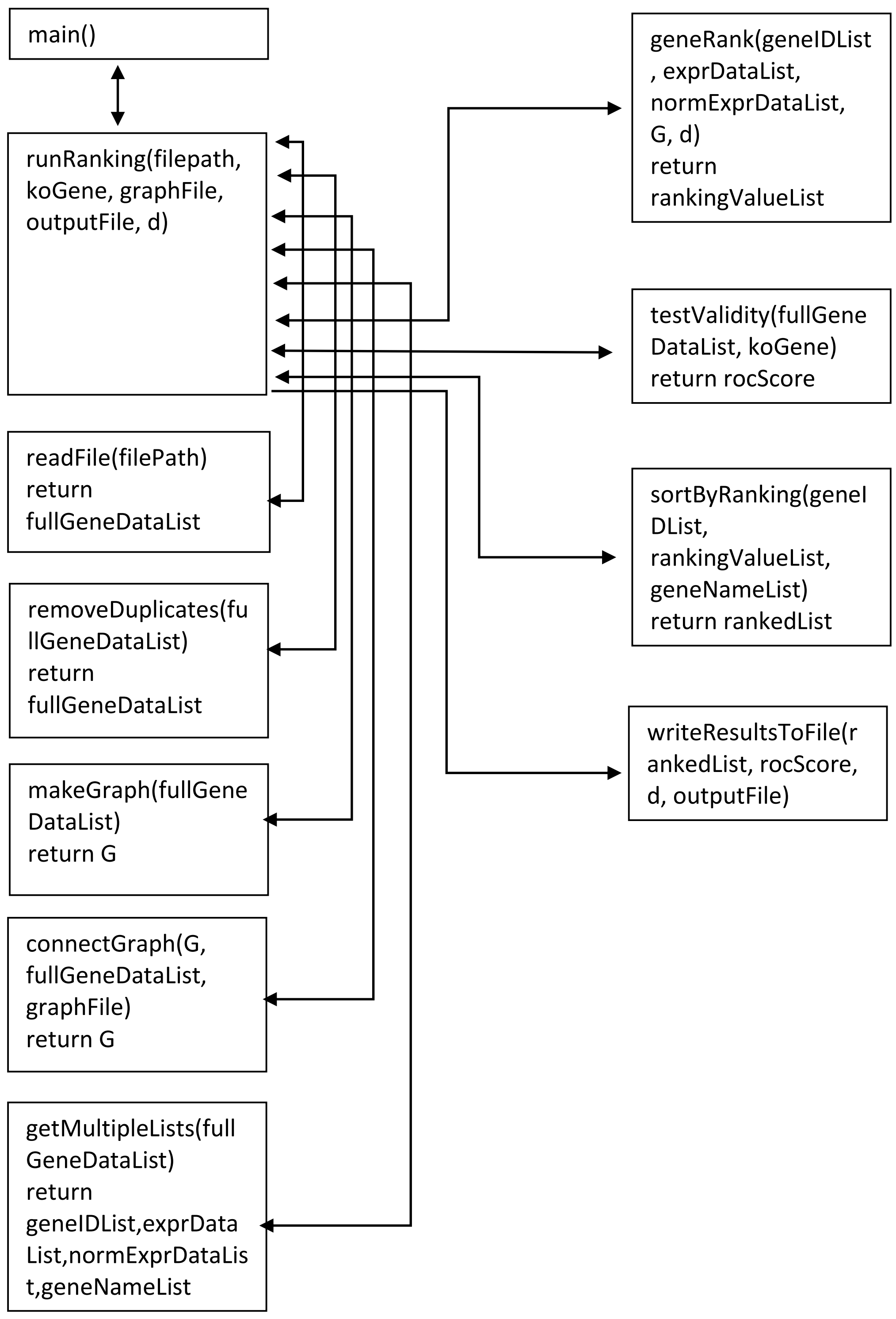

main()
runRanking(filepath, koGene, graphFile, outputFile, d)
readFile(filePath) return fullGeneDataList
removeDuplicates(fullGeneDataList) return fullGeneDataList
makeGraph(fullGeneDataList) return G
connectGraph(G, fullGeneDataList, graphFile) return G
getMultipleLists(fullGeneDataList) return geneIDList,exprDataList,normExprDataList,geneNameList
geneRank(geneIDList, exprDataList, normExprDataList, G, d) return rankingValueList
testValidity(fullGeneDataList, koGene) return rocScore
sortByRanking(geneIDList, rankingValueList, geneNameList) return rankedList
writeResultsToFile(rankedList, rocScore, d, outputFile)

Figure 5. Showing the python file intended for use by biologists. The value of d within this file is set to 0.65 as the value which is deemed most suitable for most uses of the algorithm.

5.0 Results

5.1 The goal

The aim of this study is to create a user friendly, fully accessible programme which provides the same service as GeneRank. While also, evaluating the success of GeneRank as well as the claims found within the results and conclusions by Morrison *et al.* [22]. The Success of GeneRank is then to be measured against modern techniques which use modern machine learning approaches. The results of these approaches come some six years after GeneRank and therefore are more advanced, however, they still use the underlining structure of GeneRank. In that they use the combination of expression data and connection data.

In order to fully evaluate GeneRank against these new approached, the same datasets were used, with the same measures for evaluation shown. Using these datasets and evaluation methods it is also possible to evaluate the claims founded by GeneRank. The claim that the algorithms produced are better at ranking than when only expression data is used, and that it is better than a random ranking.

5.2 General Use Of GeneRank

Shown in Figure 6 is an example output for the GeneRank algorithm which can be used by biologists. This is the standard output, it is impossible to tell just by this however, how good the ranking list performs. It is unknown as to what the best prioritised ranking list for this example gene set for this experiment is. This is shown only to illustrate what the output from the algorithm looks like in regular use. The proof and evaluation of GeneRank and how good any prioritised ranking list produced will be, follows below.

```
Final ranking list and roc information for variable d = 0.65

1416119_at (Txn1) is ranked: 1 with a ranking value of: 6.464841792793212
1419308_at (Invs) is ranked: 2 with a ranking value of: 5.577904218557109
1423476_at (Slc46a2) is ranked: 3 with a ranking value of: 5.37662583401203
1426110_a_at (Lpar1) is ranked: 4 with a ranking value of: 5.292491131792449
1456002_at (Xpa) is ranked: 5 with a ranking value of: 5.264482241338181
1450511_at (Musk) is ranked: 6 with a ranking value of: 5.191823421598711
1452122_at (AI314180) is ranked: 7 with a ranking value of: 5.117243983214608
1421059_a_at (Alg2) is ranked: 8 with a ranking value of: 5.038288714563775
1451194_at (Aldob) is ranked: 9 with a ranking value of: 5.019160045540178
1433645_at (Slc44a1) is ranked: 10 with a ranking value of: 4.775206833080054
1419009_at (Actl7a) is ranked: 11 with a ranking value of: 4.722454246552399
1423679_at (Tmem246) is ranked: 12 with a ranking value of:
4.623544106554219
1419001_at (Baat) is ranked: 13 with a ranking value of: 4.528410378647506
1425031_at (Fktn) is ranked: 14 with a ranking value of: 4.4239806775875525
1426716_at (Tdrd7) is ranked: 15 with a ranking value of: 4.350655883329503
1450392_at (Abca1) is ranked: 16 with a ranking value of: 4.274727350276168
1426857_a_at (Hsdl2) is ranked: 17 with a ranking value of: 4.20988806391586
1418294_at (Epb4.1l4b) is ranked: 18 with a ranking value of:
4.207360071807758
```

```
1456664_x_at (Gm19579///Hnrnpf) is ranked: 19 with a ranking value of:
4.207078363051547
1437026_at (Tstd2) is ranked: 20 with a ranking value of: 4.1007702854723025
1417777_at (Ptgr1) is ranked: 21 with a ranking value of: 3.974156004187761
1456644_at (Msantd3) is ranked: 22 with a ranking value of: 3.911676876282437
1452361_at (Rnf20) is ranked: 23 with a ranking value of: 3.898783185085825
1417395_at (Klf4) is ranked: 24 with a ranking value of: 3.846238144258969
1417083_at (Sec61b) is ranked: 25 with a ranking value of: 3.8170716850580004
1425287_at (Zfp189) is ranked: 26 with a ranking value of: 3.653181089758318
1451266_at (Mrpl50) is ranked: 27 with a ranking value of: 3.6093920928098546
1424083_at (Ptbp3) is ranked: 28 with a ranking value of: 3.564508628165737
1423246_at (Erp44) is ranked: 29 with a ranking value of: 3.1910719117045767
1439464_s_at (Tex10) is ranked: 30 with a ranking value of: 3.1692283083700197
1419182_at (Svep1) is ranked: 31 with a ranking value of: 3.1247207581574763
1421269_at (Ugcg) is ranked: 32 with a ranking value of: 3.1185559470999102
1420333_at (Txndc8) is ranked: 33 with a ranking value of: 2.930383861196248
1448635_at (Smc2) is ranked: 34 with a ranking value of: 2.921781837618997
1455134_at (Tmem245) is ranked: 35 with a ranking value of:
2.901014994915298
1418199_at (Hemgn) is ranked: 36 with a ranking value of: 2.8910701236860565
1448967_at (Nipsnap3b) is ranked: 37 with a ranking value of:
2.878986290064299
1424700_at (Tmem38b) is ranked: 38 with a ranking value of:
2.871820426661863
1424142_at (Ikbkap) is ranked: 39 with a ranking value of: 2.8709655105025305
1420930_s_at (Ctnnal1) is ranked: 40 with a ranking value of:
2.866449632638794
1417773_at (Nans) is ranked: 41 with a ranking value of: 2.838865972719459
1420551_at (Murc) is ranked: 42 with a ranking value of: 2.82936719412917
1433663_s_at (Ncbp1) is ranked: 43 with a ranking value of: 2.759298892513149
1427924_at (Stx17) is ranked: 44 with a ranking value of: 2.7433259952599274
1420893_a_at (Tgfbr1) is ranked: 45 with a ranking value of:
2.7344950509095143
1424222_s_at (Rad23b) is ranked: 46 with a ranking value of:
2.7082375600304456
1422689_at (Toporsl) is ranked: 47 with a ranking value of: 2.6776739504829568
1449168_a_at (Palm2///Akap2) is ranked: 48 with a ranking value of:
2.650194046862632
1424678_at (Actl7b) is ranked: 49 with a ranking value of: 2.62317902980919
1420793_at (Mup4) is ranked: 50 with a ranking value of: 2.6194502412605707
1437120_at (Snx30) is ranked: 51 with a ranking value of: 2.541376791517851
1417082_at (Anp32b) is ranked: 52 with a ranking value of: 2.492574639476444
1422754_at (Tmod1) is ranked: 53 with a ranking value of: 2.4652255805932355
1421079_at (Nr4a3) is ranked: 54 with a ranking value of: 2.3746069432685157
1425150_at (Acnat2) is ranked: 55 with a ranking value of: 2.3057999132064286
1450517_at (Tal2) is ranked: 56 with a ranking value of: 2.1898602738293116
1426649_at (Tmeff1) is ranked: 57 with a ranking value of: 2.1887215148675527
1452367_at (Coro2a) is ranked: 58 with a ranking value of: 2.1200217467890057
1450649_at (Gng10) is ranked: 59 with a ranking value of: 2.103493823168544
1423377_at (Igfbpl1) is ranked: 60 with a ranking value of: 1.9349098020302014
```

```
1448755_at (Col15a1) is ranked: 61 with a ranking value of: 1.6179882710651703

The roc score is: 0.75
```

Figure 6. Showing output produced by the file “ANGeneRank_Final” while using the “FullGONetwok” with d set to 0.65

5.3 Validation of GeneRank

In order to evaluate GeneRank as an algorithm in how well it performs in itself and against modern approaches the methods described previously have been used. The largest indication of how well the algorithm performs is the average rank of the KO gene across all 40 KO gene files, along with the top10 and top20 counts. This is, as described previously, calculated for all values of d and done repeatedly over 5 different networks. This gives a good picture of how the weighting value d influences results as well as how the network information used can also affect results. Tables 2 to 6 show the overall results for GeneRank across these 5 different networks.

Table 2. Showing an overview of results for the full GO ID network, combining Component ID, Process ID and function ID.

Network	d weighting	Average Rank	Top10 Count	Top20 Count	ROC
Full GO ID Network	d = 0.0	42.625	5	8	0.346057
	d = 0.05	41.225	6	6	0.360068
	d = 0.10	40.1	6	8	0.373553
	d = 0.15	39.05	6	10	0.389696
	d = 0.20	37.175	8	11	0.408852
	d = 0.25	36.125	9	12	0.427161
	d = 0.30	35.25	9	12	0.445652
	d = 0.35	33.775	9	12	0.463055
	d = 0.40	32.375	11	12	0.478582
	d = 0.45	31.525	11	14	0.491853
	d = 0.50	30.95	11	14	0.503923
	d = 0.55	30.175	11	15	0.51535
	d = 0.60	29.55	12	15	0.526874
	d = 0.65	28.95	12	16	0.53769
	d = 0.70	28.275	12	16	0.547359
	d = 0.75	27.65	11	16	0.555838
	d = 0.80	27.065	11	16	0.564188
	d = 0.85	26.675	11	17	0.572121
	d = 0.90	26.225	11	19	0.581774
	d = 0.95	25.125	12	20	0.598351
	D = 1.0	20.575	18	19	0.625402

Table 3. Showing an overview of results for the Component ID Network

Network	d weighting	Average Rank	Top10 Count	Top20 Count	ROC
Component ID Network	d = 0.0	42.625	5	8	0.346057
	d = 0.05	41.1	6	8	0.362051
	d = 0.10	40.2	7	10	0.375863
	d = 0.15	38.85	10	10	0.392055
	d = 0.20	37.8	10	11	0.405979
	d = 0.25	37.0	10	12	0.42149
	d = 0.30	36.0	11	12	0.436423
	d = 0.35	34.95	11	12	0.450122
	d = 0.40	34.65	11	12	0.463897
	d = 0.45	33.525	12	12	0.477639
	d = 0.50	32.7	11	13	0.490588
	d = 0.55	31.675	11	13	0.502819
	d = 0.60	30.825	10	14	0.513635
	d = 0.65	30.4	10	14	0.523197
	d = 0.70	30.025	10	16	0.531483
	d = 0.75	29.25	10	16	0.538955
	d = 0.80	28.9	10	18	0.545815
	d = 0.85	28.5	9	18	0.553619
	d = 0.90	27.9	11	18	0.563277
	d = 0.95	26.95	11	18	0.575219
	d = 1.0	21.1	16	21	0.60615

Table 4. Showing an overview of results for the Function ID Network

Network	d weighting	Average Rank	Top10 Count	Top20 Count	ROC
Function ID Network	d = 0.0	42.625	5	8	0.346057
	d = 0.05	39.925	6	9	0.36755
	d = 0.10	38.45	8	10	0.395174
	d = 0.15	37.075	9	11	0.427075
	d = 0.20	35.3	10	11	0.455048
	d = 0.25	32.85	11	14	0.478186
	d = 0.30	31.1	11	15	0.498296
	d = 0.35	30.075	11	15	0.515436
	d = 0.40	28.975	12	15	0.528996
	d = 0.45	28.1	12	17	0.540032
	d = 0.50	27.3	13	18	0.549513
	d = 0.55	26.625	13	19	0.557971
	d = 0.60	26.05	13	19	0.56496
	d = 0.65	25.925	14	19	0.571478
	d = 0.70	25.875	13	19	0.57717
	d = 0.75	25.45	13	19	0.582985
	d = 0.80	25.45	13	19	0.587991
	d = 0.85	25.075	13	19	0.594294
	d = 0.90	24.75	13	19	0.601406
	d = 0.95	24.0	14	19	0.61166
	d = 1.0	28.05	14	18	0.557976

Table 5. Showing an overview of results for the Process ID Network

Network	d weighting	Average Rank	Top10 Count	Top20 Count	ROC
Process ID Network	d = 0.0	42.625	5	8	0.346057
	d = 0.05	38.65	9	11	0.37112
	d = 0.10	38.35	8	11	0.391562
	d = 0.15	36.95	8	11	0.41985
	d = 0.20	34.2	10	12	0.448675
	d = 0.25	32.25	9	14	0.476385
	d = 0.30	30.525	11	14	0.500354
	d = 0.35	29.75	11	14	0.519091
	d = 0.40	28.575	10	15	0.534131
	d = 0.45	27.975	10	17	0.544958
	d = 0.50	27.425	11	17	0.554466
	d = 0.55	27.175	11	18	0.561573
	d = 0.60	26.775	11	18	0.568026
	d = 0.65	26.475	11	18	0.574297
	d = 0.70	26.025	11	19	0.580375
	d = 0.75	25.725	12	19	0.585531
	d = 0.80	25.475	12	19	0.590692
	d = 0.85	25.325	12	19	0.594889
	d = 0.90	24.975	12	19	0.599874
	d = 0.95	24.65	13	19	0.605432
	d = 1.0	33.15	11	13	0.539855

Table 6. Showing an overview of results for the Protein-protein interaction Network based on 38 files. This table shows both the actual and the maximum possible top10 and top20 counts. The “actual” results are based on the true values from the 38 files available. The “max” values show the maximum possible values, assuming therefore, that the two missing files would contain KO genes within the top10/top20. This allows some further conclusions to be drawn.

Network	**d weighting**	**Average Rank**	**Actual Top10 Count**	**Max Top10 Count**	**Top20 Count**	**Max Top20 Count**	**ROC**
Protein-protein interaction Network	d = 0.0	28.0	10	12	12	14	0.419102
	d = 0.05	27.816	7	9	15	17	0.416109
	d = 0.10	27.947	7	9	13	15	0.414569
	d = 0.15	28.026	7	9	13	15	0.414628
	d = 0.20	27.895	7	9	13	15	0.415009
	d = 0.25	27.868	7	9	14	16	0.417884
	d = 0.30	27.842	5	7	13	15	0.420613
	d = 0.35	27.553	5	7	13	15	0.423106
	d = 0.40	27.079	5	7	13	15	0.42626
	d = 0.45	26.816	6	8	15	17	0.42761
	d = 0.50	26.605	7	9	15	17	0.428651
	d = 0.55	26.816	7	9	14	16	0.429458
	d = 0.60	26.921	7	9	14	16	0.429561
	d = 0.65	26.895	7	9	13	15	0.430455
	d = 0.70	26.947	7	9	13	15	0.431541
	d = 0.75	26.895	7	9	13	15	0.433829
	d = 0.80	27.079	8	10	14	16	0.435839
	d = 0.85	27.079	7	9	14	16	0.439271
	d = 0.90	27.053	7	9	15	17	0.443914

	d = 0.95	26.711	8	10	15	17	0.45193
	d = 1.0	21.526	9	11	21	23	0.536628

It is important to note that for the protein-protein interactions based network only 38 of the 40 KO genes were found present and thus the average rank, top10, top20 and ROC measures are calculated over 38 and NOT 40 files. Therefore the average rank can only be used to see a trend within the protein-protein network and not be compared to the other four networks. However, the top10 and top20 counts are still useful as in actual fact the best performing measures of d = 0.90, d = 0.95 and d = 1 have a very good top20 count. Although only two gene files were missing the top10 count for the protein-protein interactions network performed considerably worse for values of d above 0.2 than the other networks. Even if we assumed that the two missing files contained KO genes ranked within the top10/top20 places as shown by the "Max" counts in the table. The ROC score is similar across all networks for each value of d but again, is better the higher the value of d, and best at d = 0.90 to d = 1.0.

5.4 Parameter d Results

To illustrate the differences across the evaluation scores for all values of d another table is made. Table 7 shows the average of each score, apart from average rank across all 5 networks. Table 8 shows these averages using the maximum top10/top20 count from the protein-protein interactions network. Average rank is not shown across all 5 networks as the protein-protein interactions network results are based on 38 and not the full 40 KO gene files. An average rank measure is shown later in table 9. The average ROC score however is shown across all 5 networks as it still shows a common trend and the ROC score for the Protein-protein interactions network differs little from the other networks, unlike the average rank.

Table 7. With the “actual” top10/20 values from the protein-protein interactions network.

d weighting	Average Top10 Count	Average Top20 Count	Average ROC
d = 0.0	6	8.8	0.360666
d = 0.05	6.8	9.8	0.375379
d = 0.10	7.2	10.4	0.390144
d = 0.15	8	11	0.408661
d = 0.20	9	11.6	0.426713
d = 0.25	9.2	13.2	0.444221
d = 0.30	9.4	13.2	0.460267
d = 0.35	9.4	13.2	0.474162
d = 0.40	9.8	13.4	0.486373
d = 0.45	10.2	15	0.496418
d = 0.50	10.6	15.4	0.505428
d = 0.55	10.6	15.8	0.513434
d = 0.60	10.6	16	0.520611
d = 0.65	10.8	16	0.527423
d = 0.70	10.6	16.6	0.533585
d = 0.75	10.6	16.6	0.539427
d = 0.80	10.8	17.2	0.544905
d = 0.85	10.4	17.4	0.550839
d = 0.90	10.8	18	0.558049
d = 0.95	11.6	18.2	0.568518
d = 1.0	13.6	18.4	0.573202

Table 8. With the “Max” top10/top20 values from the protein-protein interactions network

d weighting	Average Top10 Count	Average Top20 Count	Average ROC
d = 0.0	6.4	9.2	0.360666
d = 0.05	7.2	10.2	0.375379
d = 0.10	7.6	10.8	0.390144
d = 0.15	8.4	11.4	0.408661
d = 0.20	9.4	12	0.426713
d = 0.25	9.6	13.6	0.444221
d = 0.30	9.8	13.6	0.460267
d = 0.35	9.8	13.6	0.474162
d = 0.40	10.2	13.8	0.486373
d = 0.45	10.6	15.4	0.496418
d = 0.50	11	15.8	0.505428
d = 0.55	11	16.2	0.513434
d = 0.60	11	16.4	0.520611
d = 0.65	11.2	16.4	0.527423
d = 0.70	11	17	0.533585
d = 0.75	11	17	0.539427
d = 0.80	11.2	17.6	0.544905
d = 0.85	10.8	17.8	0.550839
d = 0.90	11.2	18.4	0.558049
d = 0.95	12	18.6	0.568518
d = 1.0	14	18.8	0.573202

The averages calculated from the true values for the protein-protein interactions network results are used are perhaps more useful. However, both tables show clearly that the average number of genes within the top 10 and within the top20 goes up linearly as d increases with few exceptions. D = 0.65 and d = 0.85 seem to be the exceptions to this rule. For both tables, the value of d = 0.65 and d = 0.85 have a higher average top10 count than the value of d below. However, they also have a slightly higher average top10 count than the next few values of d.

For the top20 count, each subsequent value of d has a higher average count than the previous value of d. This is also true for the ROC measure which shows how good the ranking process is. A ROC value of 0.5 means that the ranking is as good as a random ranking, below 0.5 shows a worse than random ranking and above 0.5 shows a better than random ranking. For values of d = 0.5 and above, the ROC value shows that the ranking is slightly better than a random ranking. It could be argued that when d is set to 0.65 or 0.85 the top10 count, the most important measure, has settled to some degree. From these two measures, when d = 0.85 the ROC score is higher than when d = 0.65, suggesting a better overall ranking.

Curiously when d = 0.65 the average top10 count is higher than the same measure when d = 0.85, this is seen in both tables. In fact d has to be set to 0.95 to beat the average top10 count found when d = 0.65, again this is found across both tables. When d is set to 0.95 or 1, the best results are found across all evaluation measures shown in the previous tables. This is also the case for the average ranking across all networks for each value of d, as shown in table 9. Table 9 shows the average ranking for the four full networks as well as for the five networks. The five networks includes protein-protein interactions which only has 38 of the 40 KO gene files available. This brings down the average ranking slightly across all values of d.

When the four full networks are used for the average ranking score for each value of d, d = 0.95 has a lower average rank. This suggests a better ranking than any other value of d as the lower the average rank the better the ranking, as the KO genes should have a low ranking. The same result is not found when the protein-protein interactions network is also included, here d = 1.0 has a lower average rank than d = 0.95. This could be a result of this network not being fully complete, with only 38 of the 40 files available.

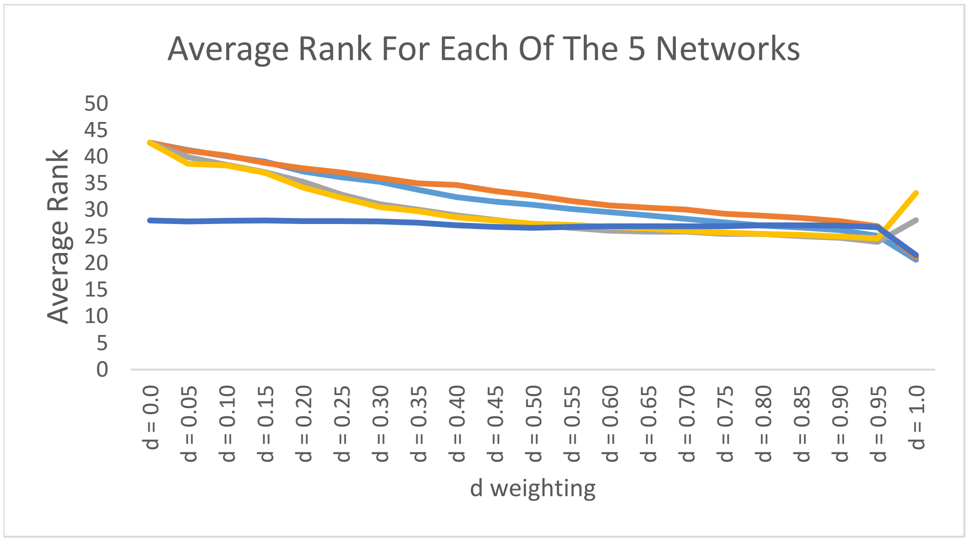
Table 9. Showing the average ranking first across the 4 full networks and then all 5 networks.

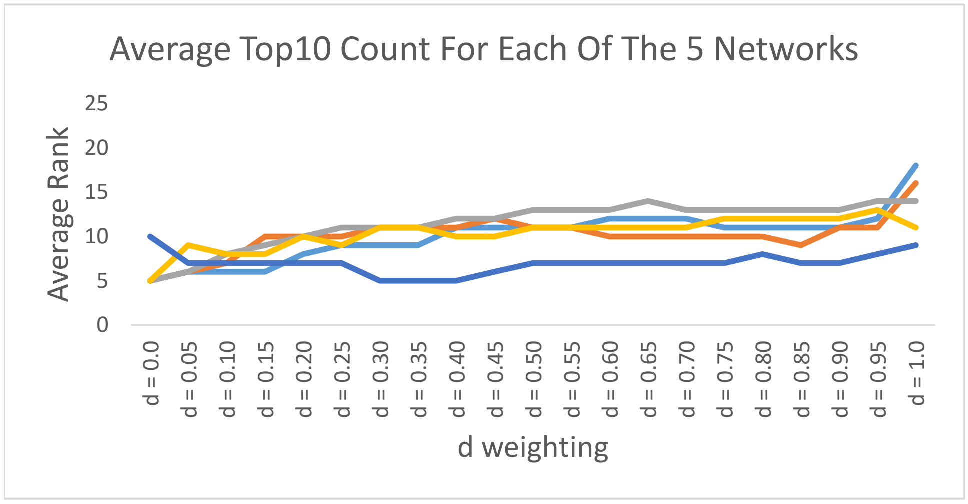
d weighting	Average Rank Across Full Networks	Average Rank Across All Networks
d = 0.0	42.625	39.7
d = 0.05	40.225	37.743
d = 0.10	39.275	37.009
d = 0.15	37.981	35.990
d = 0.20	36.119	34.47
d = 0.25	34.556	33.219
d = 0.30	33.219	32.143
d = 0.35	32.138	31.221
d = 0.40	31.144	30.331
d = 0.45	30.281	29.588
d = 0.50	29.594	28.996
d = 0.55	28.913	28.493
d = 0.60	28.3	28.0242
d = 0.65	27.938	27.729
d = 0.70	27.55	27.429
d = 0.75	27.019	26.994
d = 0.80	26.723	26.794
d = 0.85	26.394	26.531
d = 0.90	25.963	26.181
d = 0.95	25.181	25.487
d = 1.0	25.719	24.880

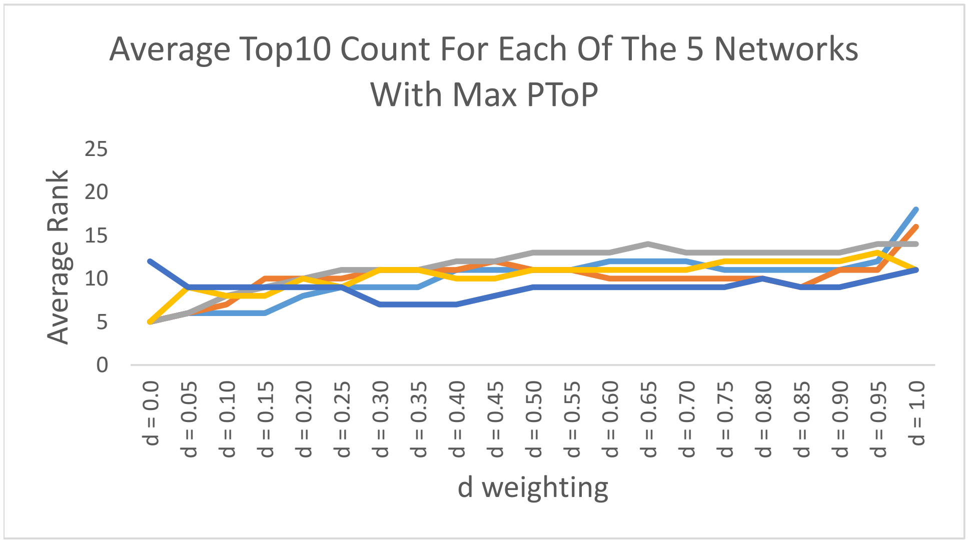
When d = 0.95 the connection data used is being considered vastly more important than the expression data and when d = 1.0 the expression data isn't used at all. The better scores for these values may be because the KO gene files contain the 100 nearest neighbours for the KO genes. The top10 count is considered to be more important than the other measures as the end goal of a ranking algorithm is to ideally rank the KO gene in the top 10 as it caused all the other changes in expression level. The top20 is still useful as a measure to show that if top10 is not guaranteed, how likely is it that it will at least be in the top20. The average ranking position is an interesting look into where you could reasonably expect each gene to be ranked across the values of d.

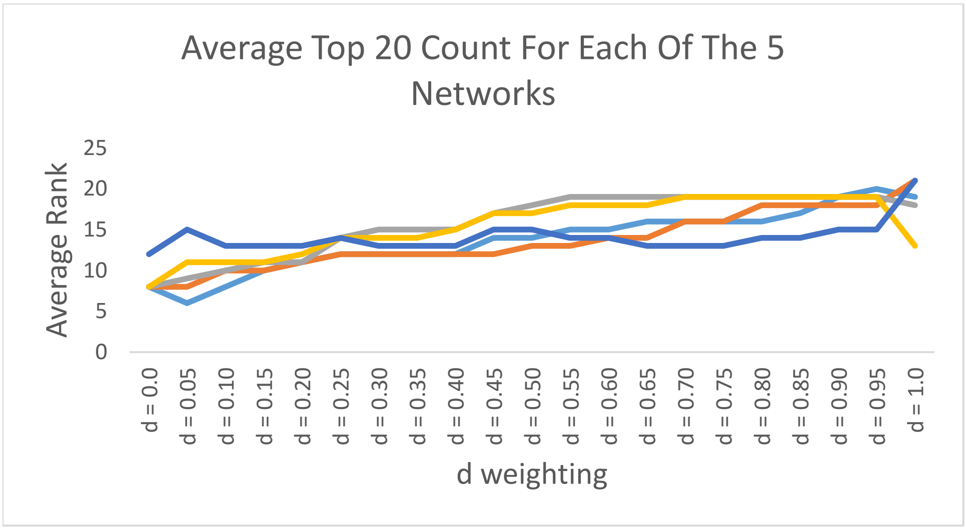
5.5 Network Comparison

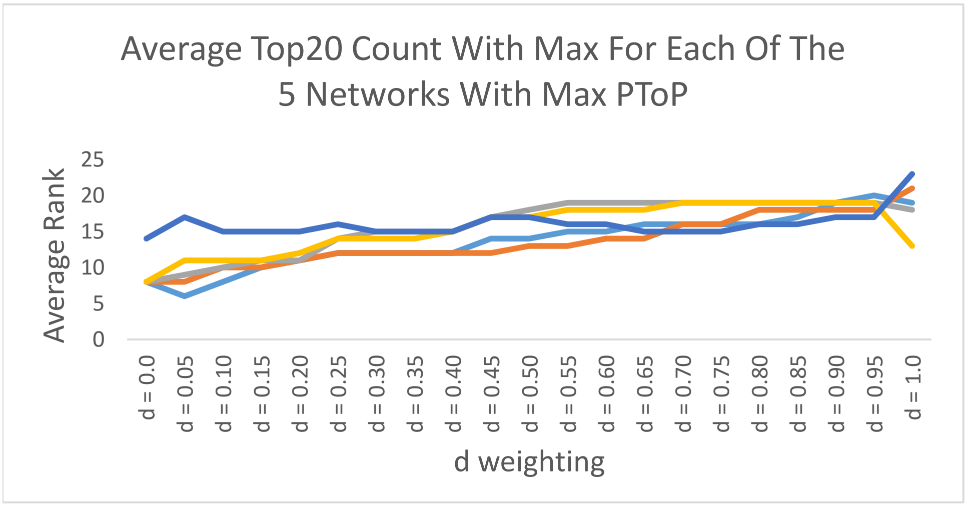
For each evaluation measure a line graph has been created showing the five networks side by side. This allows for visual comparison between networks which the tables provided above do not allow for. Figure 7 shows the following graphs in order. The average rank, the top10 count, the top10 count when using the “Max” value for protein-protein interactions network, the top20 count, the top20 count when using the “Max” value for protein-protein interactions network and finally the average ROC score. These values are the average values across the 40 KO gene files, in the case of the protein-protein interactions network, as mentioned previously, the averages are based on the 38 available KO gene files. The graphs showing the max top10 and top20 averages use the previously mentioned maximum possible count for the protein-protein interaction network.











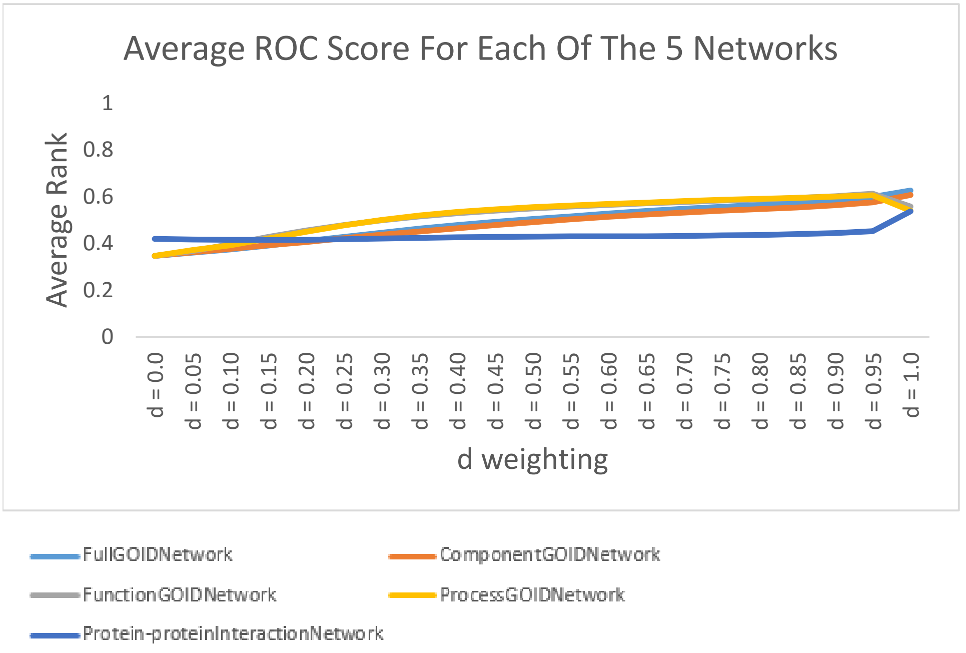


Figure 7. Showing six graphs covering all evaluation methods for each of the five networks used.

From these six graphs it is clear that the different network sources have very little effect on the overall evaluation measures. There are slight but insignificant differences in performance between the different networks used. Some networks contain many more connections than others, with the “FullGOIDNetwork” having the most connections and the protein-protein interactions networks having the least. Although the protein-protein interactions network appears to be the worst in several measures this could be because only 38 of the 40 KO gene files were available for use with this network. So the average rank and average ROC score are not directly comparable to the other networks but still show the common trends. The other three networks however, are also smaller than the “FullGOIDNetwork” and yet on some measures they outperform the larger network or are so similar there is no significant difference.

It is especially hard to say that one network performs better than the others when no network continuously outperforms the others across all four of the main evaluation measures. It is also apparent that no network performs better across all values of d, for any given evaluation measure. For different values of d, different networks appear to give better results however marginal the difference is. Some networks however, stay much more stable across measures at different values of d, for instance the protein-protein interaction network is fairly stable for the top20 count measure. There is very little change between values of d tested compared to the same measure for the other networks.

5.6 Ranking On Expression Value Only

To complete the results the average rank, top10 and top20 counts were also measured using only the expression data. The following measures are based on the absolute logFC values over the 40 KO gene files. The average rank of the KO genes was 42.625, the top10 count was 5 and the top20 count was 8.

6.0 Discussion

6.1 Validation of GeneRank

Morrison *et al.* claim that GeneRank produces a better than random ranking, that a value of 0.5 should be used for parameter d in most uses and that when d = 0 it is equivalent to ranking based only on expression change [22]. Over the five networks shown in this study the ROC score only manages to creep above 0.5 when d = 0.5. Even when d = 1, so only network information is used, the ROC score doesn’t manage to get to 0.6. This suggests that when d is set above 0.5 the ranking produced is slightly better than a random ranking. This, along with other data which shows that having a higher weighting for the parameter d gives better results across all evaluation measures puts into question the suggestion that a value of 0.5 should be used for d in most cases. A value of 0.5 is suggested for d as in their experimentation this seemed to give the most stable results.

This study would suggest that either a value of 0.65 or 0.85 should be used for d as these are the best performing values. This coincides with the apparent use of 0.85 in the PageRank algorithm used by Google which Morrison *et al.* noted when the same value of d produced the best ranking on the tests they performed [22]. The final claim is that the algorithm produces a result equivalent to ranking only on the expression change value when d is set to 0.00. This is found to be true in this study as the same algorithm has been produced, albeit in Python. For every network used in this study, when d = 0 is used the average ranking across the 40 KO gene files is 42.625.

The same value of 42.625 is produced when the average rank of the 40 KO genes is calculated based on absolute value. This means that any negative changes are converted to the equivalent positive value. So -1 would become 1. This is equivalent to the difference between the value of change and 0. It is this absolute value which is used within the GeneRank algorithm. It should be noted that this is what is meant when it is claimed that when d = 0 the algorithm produces a ranking based only on expression change values. If the true expression change values are used then a different average ranking is produced.

This study adds further proof that GeneRank does produce a better prioritised ranking list than one based only on expression change. Also proven is that when d is set above 0.5 that the prioritised ranking produced is better than an average ranking. GeneRank provided proof that combining expression data with connection data could improve the ranking methods used at the time of its creation. This improvement over ranking based only on expression data is quite considerable with the average rank falling from 42.625 to around 25 or 26 for the best performing d weightings. This initial algorithm lead the way for further improvement, with modern algorithms since 2004 being able to handle larger network structures for full genomes as well as improving the ranking again by a considerable margin [25] [29]. The best performing measures of the 2010 study by Daniela Nitsch *et al.* managed an average ranking of 8 across the 40 KO gene files [29].

6.2 Tuning Paramater d

The results of this study coincide with many of the results found by Morrison *et al.* This includes the results found across the different values used for the parameter d. This parameter is the deciding factor for how heavily connection data should be weighted against expression data. It is found within this study that when d is set to 0.65 or 0.85 the prioritised ranking list produced is better than almost all other weightings of d. Only when d is set to 0.95 or 1.0 does this change. When d is set to 1.0 the expression data is ignored and the ranking is only based on connection data. So while for the experiments conducted within this study suggest that setting d to 1.0 gives the best results this could be down to the fact that the 40 KO gene files were limited not only to just 100 genes, but to the 100 nearest neighbours to the KO gene. It could be found that these 100 genes share many more connections between each other than a random subset of 100 genes which includes the KO gene.

For use on a larger number of genes it may be found that a different value of d performs better. The justification for setting d to 0.65 or 0.85 is that for these values the top10 count was higher than both values of d above and below. As mentioned above, when d is set to 0.65 the top10 count is only beaten in many networks when d is set to 0.95 or 1.0 which is for most uses of the algorithm, too high. An average use of GeneRank would include a great number more genes and a more complex network structure as a result and so having d set to 0.95 or 1.0 is very likely to miss the aim of the experiment which is to identify the genes which have changed as the result of experimentation, e.g knocking out genes.

In this study having d set to 1.0 would mean that the KO gene information would be completely ignored, and the ranking would only be based on how connected each gene is to the others. This misses the point of the experiment which is to rank the genes according to how they changed as a result of this KO gene. Therefore having d set to anything above 0.85 would severely risk the validity of results for such an experiment. If the same experiment was done again with the 1000 nearest neighbours to the KO gene or in fact the full 22,000 genes within the mouse genome these high values of d would not be expected to have the same increased performance over lower values of d.

As a result, for most uses of GeneRank this study suggests that a value of 0.65 should be used for d, 0.85 could be used alongside this to compare the ranking lists and perhaps adjust it if desired. Or d could be set to 0.85 for experiments in which the network data is deemed more vital to the study than the expression data. Any value of d above 0.85 is likely however to lose the value of the expression change in which most experiments are based on. It is noted by Morrison *et al.* that the quality of results decrease when d is above 0.65 on one network and when d is above 0.55 for the other two networks observed [22]. In this study, for ever increasing values of d, the quality of results increases until around 0.95. As previously noted d is thought to be set to 0.85 for the PageRank algorithm in which the Google search engine uses. With all this information available, it is suggested for most cases d should be set to 0.65 or 0.85.

6.3 Network Choice and Influence

The actual choice of network appears to have very little effect on the overall results of the GeneRank algorithm during this study. It appears that as long as there is sufficient network data available, the source or network information used doesn't have a significant impact on the prioritised ranking list produced. Daniela Nitsch *et al.* found that for some ranking methods developed and used within their study that using network information from String 8.2 gave better results than when using String 7.1 [29]. They also found that this did not affect all ranking methods in a significant way. Some ranking methods appear to be influenced less by the size or confidence of the network data while than other methods. GeneRank appears, with the networks used within this study, not to be affected greatly by a change in network information.

For this study, the confidence for network connections was not used, it is possible to use the confidence measure between 0 and 1 as the edge connection between genes. For GeneRank this would mean that when two genes were not connected the Wij measure used within the algorithm would still produce a value of 0. However, if the two genes were connected instead of producing a value of 1 this value would be the confidence value. Such that is the confidence value for genes I and j being connected was 0.6 the value 0.6 would be put in the place of Wij in the algorithm.

The inclusion of the confidence value should affect the prioritised ranking list produced by the algorithm. In this study this information was not used as the confidence values for GO Ontology networks was not given by the databases used.

Only the protein-protein interactions network sources within String 10.0 provided such confidence values. This is perhaps another area for further work to see how this would affect results in a positive or negative way.

6.4 Comparison With Modern Techniques

Compared to the modern methods described by Daniela Nitsch *et al.,* GeneRank performs worse across all evaluation measures [29]. The average rank produced as well as the top10 count across the 40 KO gene files are far superior compared to that produced by GeneRank. The difference between the top20 count between GeneRank and the methods used by Daniela Nitsch *et al.* are smaller for some of the measures tested. However, the overall difference remains large. The best average rank produced by GeneRank was 20.575 at d = 1.0 using the “Full GO ID Network”. This is compared to a best average rank of 8 produced by the methods described by Daniela Nitsch *et al.* This result is also for when d = 1.0 meaning that only connection data was used by GeneRank, when d is set to 0.65 as suggested within this study the average rank produced by GeneRank rises to 25.925 which is produced when using the “Function ID Network”

The best top10 count produced by GeneRank was 18 when d was set to 1.0 using the “Full GO ID Network”. Again when the suggested value of 0.65 is used, this measure worsens. The highest top10 count for d = 0.65 is 14 when using the “Function ID Network”. The best top10 count produced by the algorithms used by Daniela Nitsch *et al.* was 32 which is a considerable improvement [29]. For the top20 count, the best results produced by GeneRank was 21 when d = 1.0 for both the “Component ID Network” and the “Protein-protein Interactions Network”. When the suggested value for d of 0.65 is used the best result falls to 19 when the “Function ID Network” is used. This is compared to a best result of 34 within the algorithms used by Daniela Nitsch *et al.* [29].

Although the improvement across these evaluation measures is considerable, with the top10 count of GeneRank being less than half that of the same measure within modern approaches this is to be expected. The algorithms produced by Daniela Nitsch *et al.* come some six years after GeneRank and focus on machine learning techniques. It is to be expected that since 2010 better algorithms will have been produced. This does not take anything away from GeneRank where the idea to combine expression data with connection data was first proposed [22]. GeneRank itself has been improved on since it was first proposed and still remains an option in gene prioritisation today as it is open sourced, well referenced, and available for all to use [25].

6.5 Limitations Of Study

This study, like most studies based on ranked gene prioritisation suffers most from the limited scale in which the experiments are conducted. Within this study, the original GeneRank study and the study on modern approaches the number of genes in which the tests are based is very small. For instance in this study the evaluation is based on the measures used by Daniela Nitsch *et al.* over 40 files of 100 genes [29]. In actual practice these algorithms are likely to be used on larger scales, instead of wanting to rank the 100 nearest neighbours of a KO mouse gene, a ranking for all ~22,000 genes may be desired.

It is hard to say based on these studies that the algorithms produced will produce a good enough prioritised ranking list over such a large number of genes. The time cost of running each algorithm will also increase, and in the case of GeneRank increase at the rate of O(n^2). Where for n number of genes, n^2 comparisons are needed for the network creation and n iterations are required which again, compares all genes to all others at a rate of n^2 comparisons. This means that as the number of genes included goes up, the time it takes to compute rises considerably.

This could be the difference between 100 genes taking a few seconds to run and 6,000 taking minutes. At this rate 22,000 genes could take too long to compute over for a reasonable study which is needed to be run multiple times for different KO genes or different expression changes measured. Within this study it is suggested that the graph produced can be viewed using Cystoscape. This is not possible for graphs which have too many connections, which is possible when 22,000 genes are to be included.

More research within prioritised gene ranking should be put into the real life uses of these algorithms. Within this study the quality, or confidence of expression data and connection data is not used. In real use within biology both the expression change data and the connection data should be vilified and included within the algorithm. As discussed previously this could be done by using the confidence value for connections as the connection value used within the GeneRank algorithm. Expression change data would need to be vilified before being used.

This study also does not include some other types of network data such as that produced by further research into gene interactions. These other networks could perhaps have an impact on the final results, although the network data sources seem to have little effect on the outcomes within this study.

6.6 Further Study

Further study should be done across all prioritised ranking algorithms into how well they perform when more genes are included in the same experiment. For GeneRank itself more study could be done into other types of networks, the different combinations of networks available. Including the different sources available within String 10.0 for protein-protein interaction networks. This could be combined with the use of the confidence value for each connection found and could improve the final prioritised ranking list produced.

The use of the absolute expression change value should also be looked into further, a version of GeneRank could be produced which can take in the actual expression change value. Otherwise genes with a negative expression change could be separated out from those with a positive change in order to produce three rankings. One based on all expression changes, using the absolute expression change value as done in this study, one based on negative expression changes and one based on positive expression changes. This would allow for an overall prioritised ranking list as well as a prioritised ranking list showing the most significantly down-regulated and most significantly up-regulated genes separately.

The end ranking for each gene could also be combined with the connection network graph produced. Such that for each node within the graph, the ranking is included alongside the gene name. This would allow for clusters of highly ranked genes within the network to be identified.

7.0 Conclusion

This study adds validity to the claims made by Morrison *et al.* on the success of the GeneRank algorithm [22]. This is done by using modern evaluation techniques used on modern ranking techniques and algorithms. Further proof is added that GeneRank produces a better than random prioritized ranking list when connection data is weighted higher than expression change data. Within the study GeneRank is also compared to subsequent algorithms including improved versions of GeneRank as well as more modern machine learning algoirhtms. It is shown that since GeneRank was proposed the idea of combining expression data with connection data has been taken and improved on by several different people and organizations [25] [29]. These modern techniques perform better than GeneRank but are still based on the fundamental principle of combining expression change data and connection data, which was proposed by Morrison *et al* [22].

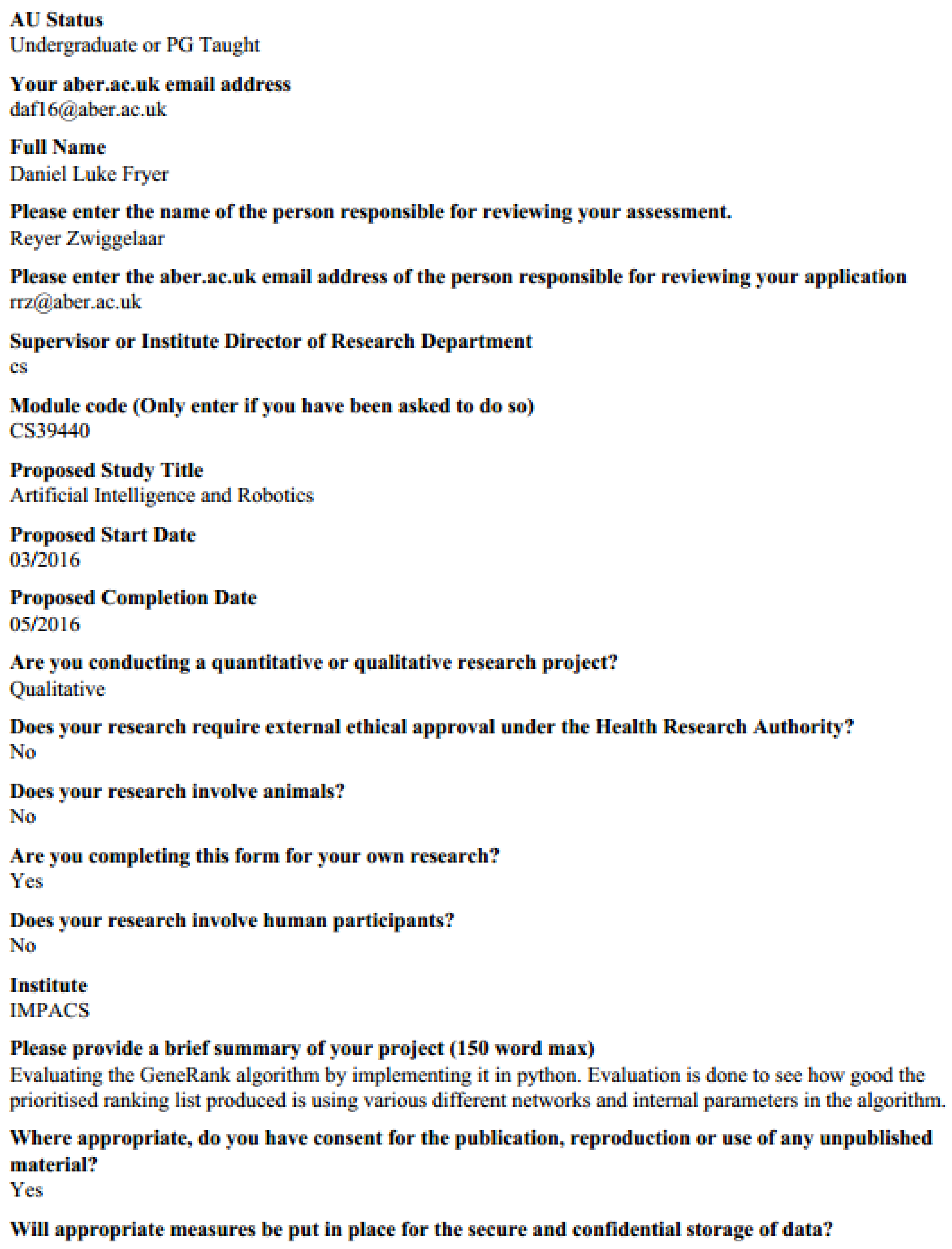
The importance of GeneRank continues as modern algorithms surpass its performance. Based on the same underlying principle modern algorithms are constantly producing better prioritized ranking lists than those used before GeneRank. These prioritized lists can be created quickly and easily by automation compared to paper based techniques used previously. These algorithms continue to use the vast quantities of data produced by biological experiments and produce results which allow biologists to target research at a few key genes. This allows time to be saved and potentially vital results to be found as research is targeted mainly at the highest ranked genes for a given experiment. This could pave the way for future discoveries into the causes and treatments of diseases.

8.0 Appendacies

A Third-party Code and Libraries

The Nextworkx library was used to create and write the network graphs from the connection information. Documentation for this can be found at
http://networkx.readthedocs.io/en/networkx-1.11/tutorial/tutorial.html
Code for sorting lists and to calculate the roc score came from the following two websites.
http://pythoncentral.io/how-to-sort-a-list-tuple-or-object-with-sorted-in-python/
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

B Ethics Submission



AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
daf16@aber.ac.uk

Full Name
Daniel Luke Fryer

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your application
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
Artificial Intelligence and Robotics

Proposed Start Date
03/2016

Proposed Completion Date
05/2016

Are you conducting a quantitative or qualitative research project?
Qualitative

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

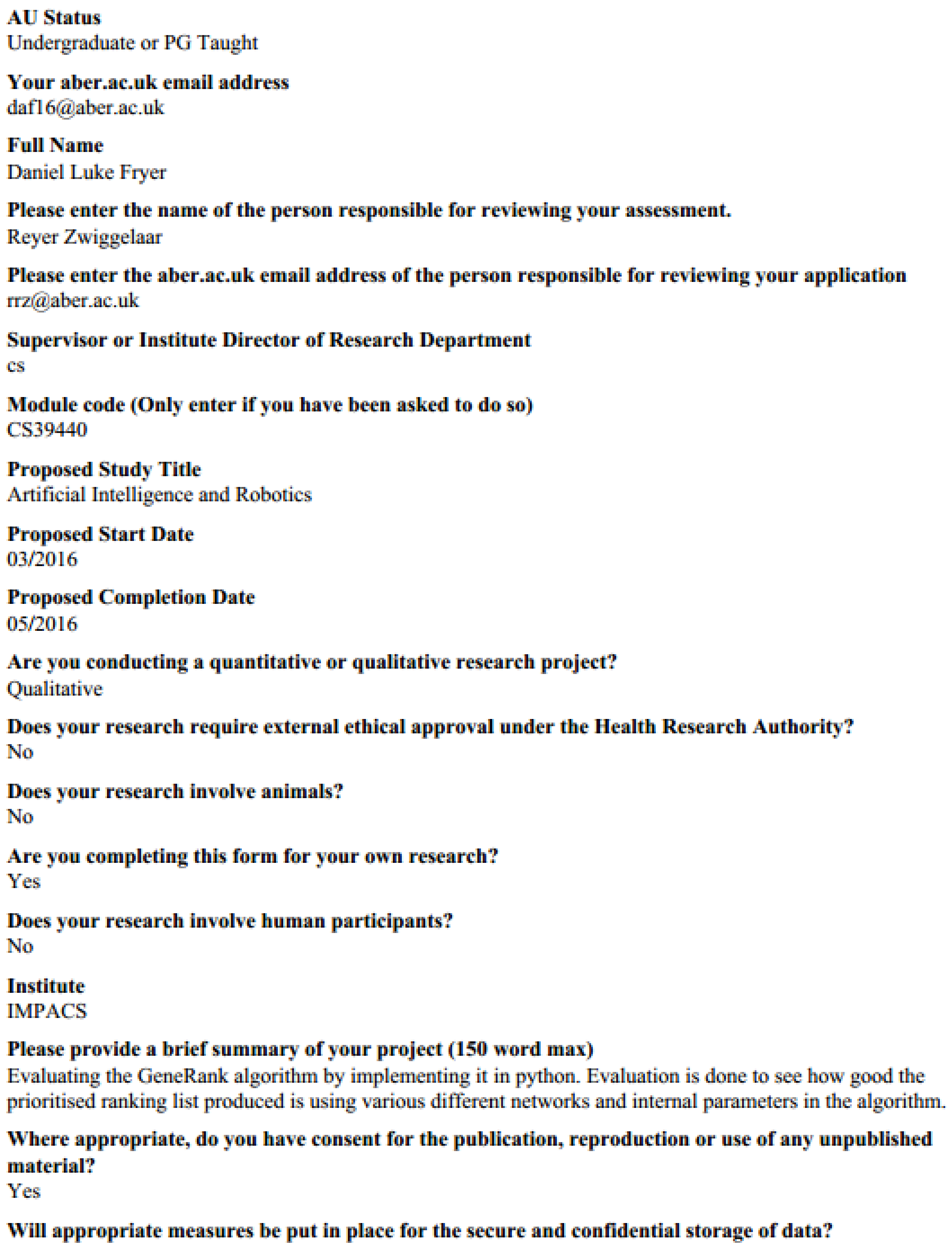
Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
Evaluating the GeneRank algorithm by implementing it in python. Evaluation is done to see how good the prioritised ranking list produced is using various different networks and internal parameters in the algorithm.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Yes

Will appropriate measures be put in place for the secure and confidential storage of data?



AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
daf16@aber.ac.uk

Full Name
Daniel Luke Fryer

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your application
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
Artificial Intelligence and Robotics

Proposed Start Date
03/2016

Proposed Completion Date
05/2016

Are you conducting a quantitative or qualitative research project?
Qualitative

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
Evaluating the GeneRank algorithm by implementing it in python. Evaluation is done to see how good the prioritised ranking list produced is using various different networks and internal parameters in the algorithm.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Yes

Will appropriate measures be put in place for the secure and confidential storage of data?

C Sample Code

GeneRank algorithm

```
# The main algorithm takes in three lists, geneIDList, exprDataList and
normExprDataList
# as well as the Graph G and parameter d. The algorithm is run such that each gene
is
# given an initial ranking score, the normalised expression value of that gene. The
ranking
# score is updated each iteration. The algorithm runs for n iterations where n is the
number
# of genes. !The data in the three lists must be in the same gene order, such that
# geneIDList[i], exprDataList[i] and normExprDataList[i] all belong to the same gene!

def geneRank(geneIDList, exprDataList, normExprDataList, G, d):
        sumOfConnectionList = []
        rankingValueList = []
        num = len(geneIDList)

        i = 0
        j = 0
        # The algorithm runs for "num" iterations, with num being the number of
        genes.
        while (i < num):
                # each gene has it's ranking updated each iteration
                while (j < num):
                        # On the first iteration add some data for each gene
                        if (i == 0):
                                sumOfConnectionList.append(0) # A value for each gene
                                which is updated after each iteration
                                rankingValueList.append(normExprDataList[j]) # initialise
                                ranking with the normalised expr value
                                # If two genes are the same then they can't be compared
                                and updated so skip to the next gene
                        elif (i != j):
                                # If the two genes are connected in graph G, hasEdge =
                                1
                                if (G.has_edge(geneIDList[i],geneIDList[j])):
                                        hasEdge = 1
                                else:
                                        hasEdge = 0
                                # temp_connection is a value based on (wij * r(j)^[i-1]) /
                                outDegree(i) or
                                # 1/0 based on connection, multiplied by the rank of gene
                                j in the previous iteration,
                                # all divided by the outDegree of gene i.
                                if (G.degree(geneIDList[i]) == 0):
```

```
                    temp_connection = (hasEdge * (rankingValueList[i-
                    1])) / 1
                else:
                    temp_connection = (hasEdge * (rankingValueList[i-
                    1])) / (G.degree(geneIDList[i]))
                # sumOfConnection[j] is the current sum of the above
                part for gene j, after i iterations
                sumOfConnectionList[j] = sumOfConnectionList[j] +
                temp_connection
                connectionValue = (1-d)*exprDataList[j]
                rankValue = connectionValue +
                d*(sumOfConnectionList[j]) # The final ranking value
                rankingValueList[j] = rankValue # update the ranking
                score for gene j

            j = j + 1
        i = i + 1
        j = 0 # reset j so that every gene is compared again next iteration

return rankingValueList # List of all genes ranking values in the order they were put
in
```

All code and documentation along with previous versions, can be found on github at https://github.com/D4nfire/Dissertation. If access is required and it is not showing then email daf16@aber.ac.uk

9.0 Bibliography

[1] D. R, "A turning point in cancer research: sequencing the human genome," *Science,* vol. 231, pp. 1055-1056, 1984.

[2] L. L. B. B. N. C. Z. M. B. J. D. K. D. K. D. M. F. W. F. R. G. D. H. K. H. A. H. J. K. L. L. J. L. R. M. P. M. K. M. J. M. J. M. C. M. W. N. J. Lander ES, "Initial sequencing and analysis of the human genome," *Nature,* vol. 409, pp. 860-921, 2001.

[3] L. H. a. L. Rowen, "The Human Genome Project: big science transforms biology and medicine," *Genome Medicine,* vol. 5, no. 79, 2013.

[4] G. Z. X.-M. Z. V. v. N. P. B. Murat Iskar, "Drug discovery in the age of systems biology: the rise of computational approaches for data integration," *Curr Opin Biotechnol,* vol. 4, pp. 609-16, 2012.

[5] C. A. Ouzounis, "Rise and Demise of Bioinformatics? Promise and Progress," *PLoS Comput Biol.,* vol. 8, no. 4, 2012.

[6] S. B. Hedges, "The origin and evolution of model organisms," *Nature Reviews Genetics,* no. 3, pp. 838-849, 2002.

[7] R. A. Ankeny, "Model Organisms as Models: Understanding the 'Lingua Franca' of the Human Genome Project," *Philosophy of Science,* vol. 68, no. 3, pp. S251-S261, 2001.

[8] J. A. B. C. J. B. J. A. K. J. E. R. a. T. M. G. D. G. Janan T. Eppig, "The Mouse Genome Database (MGD): facilitating mouse as a model for human biology and disease," *Nucleic Acids Res,* no. 43(Database issue), p. D726–D736, 2014.

[9] M. P. a. S. L. Salzbergcorresponding, "Between a chicken and a grape: estimating the number of human genes," *Genome Biol,* vol. 11, no. 5, 2010.

[10] S. A. C. a. J. M. C. David Botstein, "Yeast as a Model Organism," *Science,* vol. 277(5330), pp. 1259-1260, 1997.

[11] M. E. M. J. B. K. Andrea A. Duina, "Budding Yeast for Budding Geneticists: A Primer on the Saccharomyces cerevisiae Model System," *GENETICS,* vol. 197, no. 1, pp. 33-48, 2014.

[12] H. R. D. R. Abbott MK, "Drosophila melanogaster as a model system for assessing development under conditions of microgravity.," *Trans Kans Acad Sci,* vol. 95, no. 1-2, pp. 70-5., 1992.

[13] S. S. B. J. W. D. H. T. Plowman GD, "The protein kinases of Caenorhabditis elegans: a model for signal transduction in multicellular organisms.," *Proc Natl Acad Sci U S A.,* vol. 96, no. 24, pp. 13603-10., 1999.

[14] S. H. a. C. Creighton, "Making sense of microarray data to classify cancer," *The Pharmacogenomics Journal,* vol. 3, no. 308-311, 2003.

[15] Y. A, "Gene expression profiling for targeted cancer treatment.," *Expert Opin Drug Discov,* pp. 91-99, 2015.

[16] C. Tilstone, "DNA microarrays: Vital statistics," *Nature,* vol. 424, 2003.

[17] A. T, "Dealing with missing values in large-scale studies: microarray data imputation and beyond.," *Brief Bioinform,* vol. 2, pp. 253-64, 2010.

[18] S. M. B. J. D. K. J. R. D. a. P. N. B. David A. Orlando, "Manipulating Large-Scale Arabidopsis Microarray Expression Data: Identifying Dominant Expression Patterns and Biological Process Enrichment," *Methods Mol Biol,* vol. 533, pp. 57-77, 2014.

[19] F. J. K. J. R. J. R. M. Smith CM, "The gene expression database for mouse development (GXD): putting developmental expression information at your fingertips.," *Dev Dyn.,* vol. 10, pp. 1176-86, 2014.

[20] M. C. Langville AN, "Deeper inside PageRank," *Internet Mathematics,* pp. 1: 335-380, 2005.

[21] S. B. ,. R. M. ,. T. W. Larry Page, "The PageRank Citation Ranking: Bringing Order to the Web (1998)," 1998.

[22] R. B. D. J. H. a. D. R. G. Julie L Morrison, "GeneRank: Using search engine technology for the analysis of microarray experiments," *BMC Bioinformatics,* vol. 6, no. 233, 2005.

[23] F. T. Z. H. a. C. D. Yang Liu, "Evaluation and integration of cancer gene classifiers: identification and ranking of plausible drivers," *Sci Rep,* 2015.

[24] A. A. a. P. H. Rainer Breitling, "Graph-based iterative Group Analysis enhances microarray interpretation," *BMC Bioinformatics,* vol. 5, no. 100, 2004.

[25] E. Demidenko, "Microarray enriched gene rank," *BioData Min,* vol. 8, no. 2, 2015.

[26] S. P. K. C. C.-H. O. E. M. S. G. B. D. B. P. Gasch AP, "Genomic expression programs in the response of yeast cells to environmental changes," *Molecular Biology of the Cell,* vol. 11, p. 4241–4257, 2000.

[27] B. D, "The area above the ordinal dominance graph and the area below the receiver operating characteristic graph," *Journal of Mathematical Psychology,* no. 12, p. 387–415, 1975.

[28] R. N. Gribskov M, "Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching," *Computer and Chemistry,* no. 20, p. 25–33, 1996.

[29] J. P. G. F. O. B. d. M. a. Y. M. Daniela Nitsch, "Candidate gene prioritization by network analysis of differential expression using machine learning approaches," *BMC Bioinformatics,* vol. 11, no. 460, 2010.

[30] "DOWNLOAD ANACONDA," Continuum Analytics, [Online]. Available: https://www.continuum.io/downloads. [Accessed 10 04 2016].

[31] M. A. O. O. B. N. W. J. R. D. A. N. S. B. I. T. Shannon P, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome Res,* no. 11, pp. 2498-504, Nov 2003.

[32] E. D. B. S. C. S. B. L. M. M. F. Francesca Veronesi, "Novel therapeutic targets in osteoarthritis: Narrative review on knock-out genes involved in disease development in mouse animal models," *Cytotherapy,* vol. 5, pp. 593-612, 2016.

[33] H. H. Prossnitz ER, "What have we learned about GPER function in physiology and disease from knockout mice?," *J Steroid Biochem Mol Biol.,* vol. 153, pp. 114-26, 2015.

[34] O. O. S. B. S. A. Ideker, "Discovering regulatory and signalling circuits in molecular interaction networks.," *Bioinformatics,* vol. 18, pp. S233-40, 2002.

[35] S. T. a. H. W. Mewes, "Functional modules by relating protein interaction networks and gene expression," *Nucleic Acids Res,* vol. 31, p. 6283–6289, 2003.

[36] "Ontology Documentation," [Online]. Available: http://geneontology.org/page/ontology-documentation. [Accessed 16 04 2016].

[37] E. M. P. a. S. Fields, "Protein-protein interactions: methods for detection and analysis.," *Microbiol Rev,* vol. 59, no. 1, p. 94–123., 1995.

[38] "String 10.0," String Consortium 2016, 2016. [Online]. Available: http://string-db.org/. [Accessed April 2016].

[39] S. T. B. A. F. K. V. J. Rung J, "Building and analysing genome-wide gene disruption networks.," *Bioinformatics,* vol. 18, no. Suppl 2, pp. S202-10., 2002.

[40] M. E. O. L. C. R. J. M. J. F.-d.-C. a. R. B. Alexander Martin, "BisoGenet: a new tool for gene network building, visualization and analysis," *BMC Bioinformatics,* vol. 11, no. 91, 2010.

[41] K. G. K. S. R. J. A. D. K. T. R. P. J. B. H. V. R. F. N. M. W. W. B. M. S. H. S. U. F. H. B. M. S. H. S. M. P. C. G. R. Winter C, "Google goes cancer: improving outcome prediction for cancer patients by network-based ranking of marker genes.," *PLoS Comput Biol.,* vol. 8, no. 5, 2012.

[42] S. K. Zuber V, "Gene ranking and biomarker discovery under correlation.," *Bioinformatics,* vol. 25, no. 20, pp. 2700-7, 2009.

[43] S. K. Opgen-Rhein R, "Accurate ranking of differentially expressed genes by a distribution-free shrinkage approach.," *Stat Appl Genet Mol Biol.,* no. Article9, 2007.

[44] G. J. O. F. d. M. B. M. Y. Nitsch D1, "Candidate gene prioritization by network analysis of differential expression using machine learning approaches.," *BMC Bioinformatics.,* vol. 11, no. 460, 2010.

[45] S. L. a. R. S. Patrik D’haeseleer, "Candidate gene prioritization by network analysis of differential expression using machine learning approaches," *Bioinformatics,* vol. 16, no. 8, pp. 707-726., 2000.

[46] D. S. a. P. S. Aric Hagberg, "Networkx on Github," [Online]. Available: https://github.com/networkx/networkx. [Accessed 02-05 2016].

[47] F. a. V. G. a. G. A. a. M. V. Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[48] "download-annotations," Gene Ontology Consortium, 2016. [Online]. Available: http://geneontology.org/page/download-annotations. [Accessed March, April and May 2016].

[49] T. S. community., "Scipy.io.loadmat," The Scipy community., 20 February 2016. [Online]. Available: http://docs.scipy.org/doc/scipy/reference/generated/scipy.io.loadmat.html. [Accessed March 2016].