。态uang'tai﷽﷽﷽﷽﷽﷽﷽﷽的预测细胞内基因表达的程度on酵母基因调控网络对外界环境刺激的应对系统研究

王兵

摘要：

基因调控网络(Genetic Regulatory Network)有助于我们从系统的角度出发理解生物体。我们通过对7388份酵母微阵列表达谱数据的分析，构建了的酿酒酵母(Saccharomyces Serevisiae)的基因调控网络。在建模过程中采用三状态变化系统模型代替常用的布尔模型，并使用了广义贝叶斯算法估计马尔科夫链状态转换矩。结合基因分类注释(Gene Ontology)以及蛋白质相互作用数据等先验知识辅助推断网络模型，并交叉验证的方法验证网络的可靠性。最后使用了该网络研究酵母网络的模块性和鲁棒性及供求原则。

关键词:

基因调控网络，网络模体，生物回路，最优回路设计

Abstract：

Genetic Regulatory Network (GRN) is a systematic view of organism, which shows a great value on our understanding of evolution. In this study, we tried to module Saccharomyces Serevisiae’s GRN in a new way, which use improved bool module, Bayes algorithm and Markova chain. We collect 7388 microarray data as well as the known evidences such as protein-protein interaction data and Gene Ontology to help with the modeling process. And we evaluate the module’s accuracy by compare with the known data. We also analyzed the module’s modularity, robustness and demanding-supply attribute.

keywords：

Genetic Regulatory Network, Network modular, biological circuit, best circuit design

1. 选题背景：

进化的本质是生物体适应环境并将基因遗传传播开来。其中生物体如何适应环境又是其中最关键的问题。进化学家们相信基因的表达策略是生物体适应环境的关键。近年随着技术的发展，生物学家们使用基因表达的数量性状座位作图法(eQTL mapping)来揭示基因表达的差异的进化[1]；讨论全基因组加倍对物种适应能力的影响以及物种形成的影响[2]；还有优势突变的概率分布[3]等。这些都是从相对宏观的角度讨论种群的适应策略，对于个体的适应策略的研究则不多。同时，这些研究在机理方面的说明不足，只能给我们一个大致的方向，而各种机制则不明晰。于是我们有了系统生物学。

系统生物学家的终极理想是解读生物体，甚至模拟生物体，即精确地计算出环境与基因组的输入对时间的输出函数。我们将这个函数称之为系统。现今已有许多基于模型的系统比如布尔网络(Boolean Networks)[8]，有向图网络(directed network)[7]，贝叶斯网络(Bayesian Network)[9]以及动态贝叶斯网络[10]，微分方程(Ordinary Differential Equation)[11]，以及考虑了随机变量的Petri网络[12]等。

本文采用的系统是酿酒酵母(Saccrharmoyces Serevisiae)因对环境变化的应激系统，因为酵母是一种相对简单的单细胞生物，由于是真核生物，因此在后继的研究中应用价值更大。而且酵母也是一种工业用菌，因此研究酵母的应激(stress response)有利于提高生产效率(比如研究酵母对酒精的耐受性等)。此外，酿酒酵母作为一个模式生物，其数据库也比较全，最有名的比如SGD(Saccharomyces Genome Datbase)[4], 还有搜集分析了数千份酵母表达谱(Microarray Expression Data)的SPELL[5]，以及专门研究位点的顺序的YGOB(Yeast Gene Order Browser)[6]等。由于酿酒酵母的数据库完整，研究应用价值大，因此选择酿酒酵母的应激系统作为对象研究"性价比"较高。

2. 方案论证：

以酵母的基因调控网络(Gene Regulatory Network, GRN)作为系统进行研究的例子有很多。比如研究酵母乳糖代谢的系统[13],酵母Uri1p启动子调控系统[15]，酵母MAPK信号转导系统[16]等。此外研究酵母应激系统的例子比如Castell等研究的酵母在受到热激(Heat Shock)时基因转录效率和mRNA稳定性[14]。以及Venancio等研究的酵母系统在自然环境下进化出来的对化学药品的抵抗性[17]。这些研究为我们更大范围地理解酵母的应激系统提供了宝贵的经验和前期数据。

确立了研究对象和内容之后，我们需要确立目标，然后根据目标确立一个有效的模型。我们的目标是根据细胞的基因型和环境输入，得到一个表现型的输出。在这里我们的基因型指的是基因是否存在，即只是0或者1的状态。而对于本论文来说，特指酿酒酵母典型的实验种(S288C)的基因及其相互作用。关于环境输入，我们的期望是输入一个大致的环境描述，然后根据环境感知基因(Sensor gene)的表达变化推测剩余基因组的表达变化，即表达谱预测。另外，我们希望能在验证系统的前提下进行系统模型的鲁棒性分析和供求分析。鲁棒性原理在进化界是指“系统的某些特征在外界环境不确定的情况下保持一定的确定性”。比如外环境和内稳态的关系。而需求分析在本文指的是酵母对N源和C源的需求分析。

确定了模型的目标之后我们需要确定模型的精度与规模，原则上我们希望模型的精度越高越好，准确度也越高越好，但是，提高精度就意味着需要更多的数据来推断参数，复杂的如量子力学/分子力学(Quantum mechanics molecular mechanics, QM/MM)在蛋白质对接模型中的应用[18]。即便如此也只是一个对量子力学的近似，但这个级别的精度已经是在单个蛋白质这种规模系统下最大的精度了。在更大的尺度下，比如代谢网络水平，要精确地估计蛋白质互相作用水平已经是不可能了，而在这个尺度下我们关心的问题也不是那个特定的蛋白质和另一个蛋白质互相作用，而是比如某一类分子的浓度变化曲线，因此在这个规模的系统下，使用微分方程组(Ordinary Differential Equation)可能是更好的选择，比如用米氏方程(Michaelis-Menten Equation)及希尔方程(Hill Equation)来估计每一个化学反应，因此估计每一个反应也至少需要2个常数(结合常数KON与解离常数KOFF)。如此对于一个比较小的试管系统仍然有效，因为可以人为地控制化合物的总类。但当系统为一个活体细胞的时候就很困难了，即使研究一个非常小的系统，比如酵母半乳糖代谢系统，酶本身的浓度就不是守恒的，蛋白酶的浓度也受更大的系统调控，因此要精确地估计底物浓度变化曲线以及基因表达水平是非常困难的。此外，由于生物体本身的多样性，个体的表达水平其实并不是我们关注的最多的，相反的，我们更多的关注的是整一个种群的基因表达水平。解决方案是加入随机变量并给与一个种群数量进行模拟[19]。而对于本文的目标，系统的规模更大，是整个细胞所有基因的互相作用，因此我们能得到的精度只是基因表达的程度，比如某个基因过表达，或者某个基因低表达。

确定了模型的精度和规模，我提出了构建这个模型的6个阶段。如图1：在第一阶段我们构建的模型只具有相互连接的关系；第二阶段，模型中，每一条连接有了正相关和负相关；第三阶段，每一个相互作用明确了方向，有了明确的调控关系；第四阶段，增加了与节点(AND)，即调控关系之间的依赖性；第五阶段，即使确定了调控关系，调控发生的概率还是非百分之一百的，在这个阶段，增加了调控概率；第六阶段，调控关系需要在特定的环境下才成立，模型需要这个特定的环境的输入。该模型即是本论文的指导方案。我们的假设是，如果我们能够精确地挖掘出所有第六阶段的网络结构，我们的模型就可以精确的预测细胞内基因的时序表达状态。

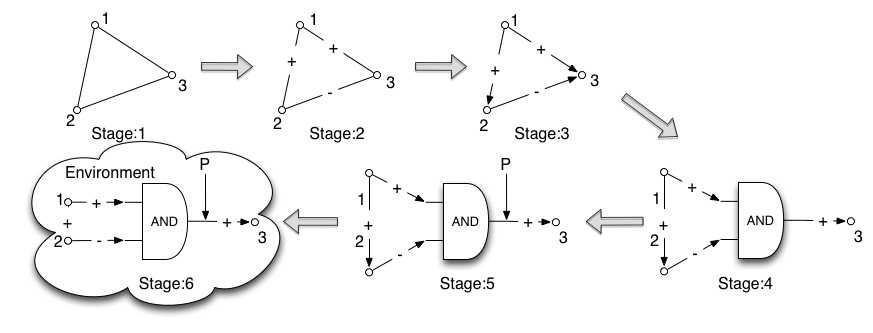


图1：系统6阶段模型

3. 过程设计论述：

3.1 系统构建综述

系统构建是所有后续计算的开始，上面的背景中已经介绍了各种精度和规模的模型及其应用范围，我的模型的基础是布尔模型与有向图模型的结合，并在布尔模型的基础上增加了第三种状态即正常表达状态(common expression status)，此外，我们主要的数据是酵母微阵列表达谱(microarray)，因此我们使用的训练数据集，以及验证用的数据集都是一个种群的平均表达量。但我们构建的系统是单细胞系统，因此我们在这个模型之上增加了种群数和随机变量。以更精确地估计生物体的平均表达水平。

此外，由于本文的主要目标是构建一个可以被验证并且可以作出一些预测的模型，而非模型本身的挖掘算法，因此我们尝试利用所有可以利用的数据，包括酵母蛋白质-蛋白质互相作用数据(interaction data)，酵母基因分类注释文件(Gene Ontology file),酵母基因Slim Gene Ontology文件，酵母单基因突变库表现型数据库(phenotype data)，酵母特征文件(SGD feature)。以及酵母表达谱SPELL数据库搜集的2394份微阵列表达谱数据和SGD上新增加的约4000份微阵列表达谱数据。

在系统模型的构建上，我们采用的数据结构是：1）节点(node)即基因，其表达状态有三种，正常表达(normal expression)，过低表达(low expression)和过高表达(overexpression)，其代表值为0，2，1。语义上过低表达为“-1”较容易理解，计算上，为了移除“-1”的符号位，用2来代替。2）边(edge)即相互作用，在这里我们仅指蛋白质-蛋白质，蛋白质-基因的物理相互作用(Physical Interaction)和基因间遗传性的相互作用(Genetic Interaction)。3）条件性函数，为基因转换状态的条件，其表现形式例如基因1高表达以及基因2低表达引起基因0正常表达。图2中描述了一个三节点的简单网络。边上的数字代表节点状态，改图表示如果基因1高表达，基因2低表达同时出现，则基因0正常表达。

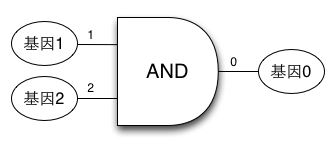


图2:一个简单的三节点模型

模型的构建方法：综上所述，我们需要构建一个条件式输入系统，即给定当前状态Z(n1,n2,n3…)，根据每个节点的条件映射函数是否成立确定节点的状态是否变化,从而得到一个新的状态向量Z’(n’1,n’2,n’3)，即Z’ = f(Z)。而本文最主要的工作是构建映射函数f。函数的表达形式是逻辑表达式堆，每一个节点有三类逻辑表达式组，即状态0逻辑表达式组，状态1逻辑表达式组和状态2逻辑表达式组。因此整个映射函数的数据结构为一个二维字典列表，列表项为一个逻辑表达式:

f\_dict[gene\_name][status] = [(condition1),(condition2),…]

问题于是被转化为计算一堆条件式，我们设计的方案是计算每两个基因和每三个基因的表达相关性，例如在95%的情况下，基因1在状态a下，基因2为状态b。则认为基因1作为基因2的输入函数之一。

下面是技术细节：

3.2 数据收集

我们能直接得到的数据来源是酵母基因组数据库(Saccharomyces Senome Database, SGD[4] )。本文所使用的数据库包括酵母蛋白质-蛋白质互相作用数据(interaction data)，酵母基因分类注释文件(Gene Ontology file),酵母基因Slim Gene Ontology文件，酵母单基因突变库表现型数据库(phenotype data)，酵母特征文件(SGD feature)。以及7388份酵母的微阵列表达谱数据。

蛋白质数据库的原始数据为tab分割的数据，用sqlite3构建SQL数据库ID2Interact，数据库结构为(SGDID\_1，SGDID\_2，InterType，ExpType，Phenotype)，SGDID为SGD数据库中基因的索引，InterType为基因相互作用类型，只有两类，即物理性作用(Physical Interaction)和遗传性作用(Genetic Interaction)。ExpType为实验类型，物理性的比如酵母双杂交(Two-hybrid)，重构复合体(Reconstituted Complex)等。遗传性的比如剂量致死(Dosage Rescue)，剂量缺陷(Dosage Growth Defect)等。构建该数据库的目的是作为检查系统输入函数中条件表达式的搜索范围。

酵母基因分类注释文件和酵母slim分类注释文件用于构建ID2GO数据库。数据库结构为(SGDID，GOID，slim，GO\_Term，GO\_Aspect，GO\_Definition),其中SGDID为SGD数据库中基因的索引，GOID为该基因的Gene Ontology注释号，slim为该GO是否是slim GO，为一个布尔值。所谓slim GO是指一个并不太大的基因分类，比如Process或者Function，也不是太小的基因分类，比如复合物结合中介(mediator complex assembly)。GO Term为该GO的条目，即一个简短的单词或词组描述的基因分类。GO Aspect为基因分类注释大类，一共有三个可能的值，分别为功能F，进程P，元件C(Function，Process，Component) 该分类最后将在网络可视化中分化为三个网络。GO Definition为该GO条目的详细定义。构建该数据库的目的是在网络可视化过程中对基因进行分类，以GO级别对基因组进行显示。

酵母单基因突变表现型数据为tab分割文件，用于构建ID2Phenotype数据库。数据结构为(SGDID，mutant\_type，phenotype，chemical，condition，details，reporter)其中SGDID依旧为基因的SGD数据库索引，mutant\_type为该基因突变类型，比如说过表达(overexpression)，或者删除突变(null)，phenotype为表现型，比如化学药物抗性(resistance to chemicals)等，后面的chemical，condition，details，reporter为选择属性，可以为空，比如表现型位化学药物抗性时，chemical属性不为空。同时，我们过滤掉了比较泛的表现型描述，比如viable。该数据库用做单突变数据检验模型的有效性。

酵母特征文件为酵母基因的各个特征的集合，用于构建ID2Feature数据库。数据库结构为(SGDID，feature，standard，Type，Descrip)。其中SGDID依旧为SGD索引，feature为基因的feature名，比如GAL1,standard为基因的系统命名，比如YBR020W，Type为基因的类型，多数为开放阅读框(Open Reading Frame, ORF),Descrip为基因的描述。该数据库用来做各种基因重名之间的互换，用于产生字典数据结构的other2ID，一个基因的所有的命名包括SGDID以及各种别名(alias)都经过该字典转换成唯一的SGDID。

酵母产物数据用于构建ID2Product数据库，数据结构为(SGDID，product)。其中基因产物为product，该数据库用于人工微调系统参考。

酵母微阵列数据比较复杂，形式比较多，数据量也比较庞大，总计8133个microarray数据(有冗余)。比较复杂是指，基因命名不一样，使用的芯片类型不一样，数据参考值不一样，条件描述不一样等。下面是具体的处理方法。

对于基因命名不一样，我们的处理方法是使用上文提到的other2ID字典进行转换，转换成唯一对应的SGDID。构建一个基因列表，使用SGDID作为索引，对所有的微阵列数据扫描一遍后，寻找出所有出现过的基因并放入列表。

对于芯片的数据参考值不一样，即有一部分芯片采用相对值。我们的处理方法是参考SPELL数据库构建的方法[5]：首先对所有的值扫描一遍，如果均值在0左右的说明已经经过log转换，如果均值远大于0(比如2.xx)则认为是没有经过log转换的。但后来发现即使经过log转换，均值依旧可能接近1，而另一方面，原始数据也可能是接近0的均值。后来追加了一条规则，如果微阵列有值小于0的值的位置，则认为整个芯片已经经过对数转换，反之则对每一个基因的各个条件表达取平均，然后对每一个值取以二为底的对数作为相对值。

接下来，比较麻烦的问题是，如何将所有的数据读入内存以供运算。最初的解决方法是直接将数据存入以基因，CID为2维索引，以表达水平值为值的字典。(注CID是指Condition ID，每一个芯片有一个实验条件，即为Condition，该实验条件在论文中出现的顺次为Index，一个CID由两部分组成，PMID+Condition\_Index，PMID即论文的PubMed索引，因此CID可以唯一代表一个实验条件。这个解决方案最后失败了，因为32位的Python无法分配足够的内存给这个字典。方案二是构建一个SQL数据库，数据库结构为：microarray(SGDID，CID，EXP，EXP\_bool)。EXP是经过log转换的表达水平。EXP\_bool则是更简单的表达水平布尔值，虽然定义上可以取的值有四种，0：正常表达；1：过表达；2：低表达；99：数据不明。其中对过表达和低表达的定义分别为取对数之后表达值高于2和低于-2。即实际表达量高于正常水平的四倍为过表达以及低于四分之一的为低表达。方案二很好的解决了内存的问题。但仍旧有一个非常大的问题是，在下文中会描述的计算复杂度的问题，整体计算复杂度是O(n2)级别的，其中n还不是指节点数，而是指所有节点的相互作用节点数(原来的算法是O(n3)的，某个程度上已经优化了很多了，下文详述)。在这个运算量的前提下，将数据存储在数据库中无法满足需求。因此有了方案三：方案三是利用Python的插件numpy构建可修改值的数组列表。这个列表有一个缺点是，必须在初始值的时候给以确定的列表大小。用纯数组构建列表，然后横坐标的索引是SGDID，纵坐标的索引是CID，分别有一个字典对应索引。完成数组的构建，这样大约需要近500M左右的内存。

以上是数据库的构建过程，图3是数据库的关系模型（ER图）。

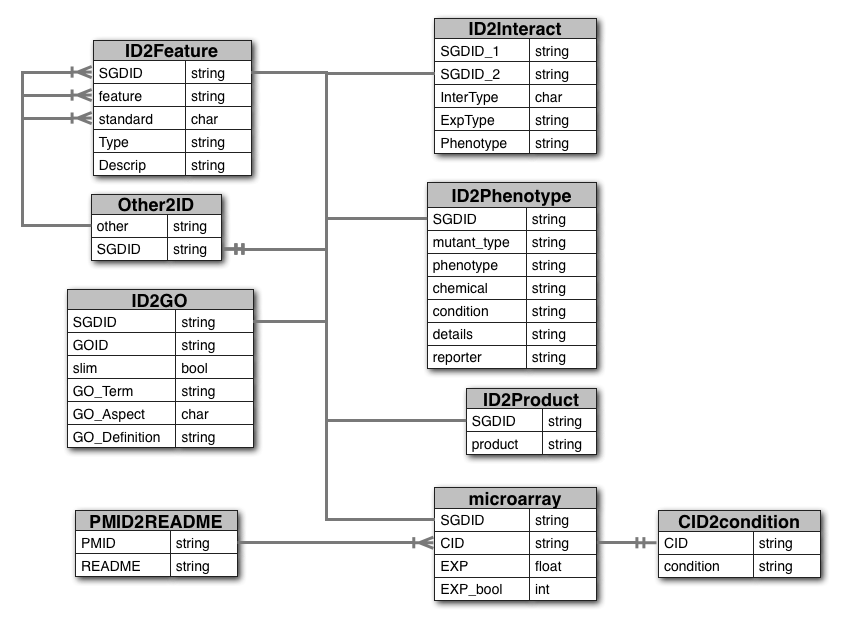


图3：数据库ER图

3.3 网格计算及网络整合

计算条件函数是整个系统的核心，如上文所述，我们的目标是计算每个基因的转换条件函数。这个函数是由逻辑语句构成的。例如：

S000000421 2 AND S000005668 2 => S000004660 2

S000001533 1 => S000000440 0

这两个例子一个是双基因并输入，另一个是单基因输入。基因名以SGDID表示。于是在这一步骤中，我们想要得到的数据就是一张巨大的表，包含所有基因的转换语句。

方案一：表达谱中所有涉及到的基因有5939个，表达谱一共有8133张表达谱数据，去冗余后也有7388张。使用线形相关性分析任意两个节点的相关性后构造单输入基因语句，分析任意三个节点的相关性后构造双输入基因语句。这个方案原理上可行，但实际上存在三个问题：1）分析任意两个节点的相关性后构造单输入语句。首先只有两个节点的调控关系不多，通常都是层次调控的节点，节点多具有多输入，多输出的特征，即使有这样的单输入单输出的节点，我们也无法从相关性中推知因果性，即调控关系还需要进一步推断。2），三个节点的相关性比较难推断，因为无法用线性模型估计三个节点的相关性。只能用非线性模型，这个意味着更巨大的计算量。3），算法的时间复杂度是在O(n3)的级别，考虑到节点数在6000的级别，搜索空间在8000的级别，估计不可行。

方案二：使用基因间互相作用数据减少搜索空间。使用贝叶斯公式估计调控关系，比如在基因A=S1的情况下，观察到B=S2的概率是P(B=S2|A=S1)，又先前计算过的B=S2的单独概率是P(B=S2)，A=S1发生的单独概率是P(A=S1)。则观察到B=S2的情况下A=S1的概率为：

虽然我们也可以直接观察B=S2的情况下A=S1的概率，但是搜索P(A=S1)和P(B=S2)要快得多时间上几乎可以忽略不计，这样可以将计算量缩减一半。三个节点的相关关系P(A=S1|B=S2 AND C=S3)同理可得。此外，我们只考虑有物理上直接相互作用的两个基因，或者遗传上直接相互相关的两个基因之间存在调控关系，虽然可能损失一部分的精度，但是在运算量上却减少了很多，几乎可以说从O(N3)降低到O(N2)的复杂度。不过即使如此，在试运行过程中，2.4GHz Core i5的处理器8G，DDR3内存显示的需要的时间是8000个小时左右，基本上需要跑一年，即使使用实验室12×2核的处理器的服务器也需要近一个月左右，该方案需要改进。

方案三：在方案二的基础上，增加了一个判定条件，即如果P(A=S1|B=S2)成立的概率超过95%，则不考虑P(A=S1|B=S2，C=?)的概率。这里会损失一部分信息，尤其是当C基因可以抑制B基因对A基因的调控作用时。但是考虑到1）C基因对B基因产生抑制时概率P < 95%会被过滤(下文关于置信度和P\_value的讨论中会涉及)而概率P > 95%时认为C基因对B基因的影响忽略不计。这一步可以将部分计算从O(n2)级别降低到O(n)级从而大大提高运算效率。此外，为了可以在实验室的服务器上运行程序，需要将程序并行化计算。算法本身是可以并行化的，共享的部分就是Microarray表，只需要将其指向同一块内存即可，甚至可以每一个进程都使用独立的Microarray表。另外在程序优化上，也考虑过用C++改写代码或者部分代码，但考虑到对C++并不熟悉，尤其是Sqlite的API。而改写部分代码时发现Python与C++的接口模块无法转换Python Dict数据结构。基于这个原因，我们最后只使用了1/7的数据，也就是1000个Microarray来构建模型。这样整体的速度提升了7倍，在实验室的服务器上使用10个核差不多用了两天。

关于条件函数的置信度和P\_value：本实验选采用的置信度为95%，即只当P(A=S1|B=S2) > 95%时才考虑下一步P\_value的验证，本实验中P\_value的计算公式是：

公式中，p为P(A=S1)的概率，n为样本量，即B=S2的数量，x为A=S1出现的数量。该公式计算的是A=S1出现x次及更多次情况下的概率积分。在本论文中，该公式由于计算复杂，直接计算会耗费大量的CPU,我们采取了更简便的方式，即直接根据P(A=S1)反推出P\_value < 10-7时n的截断值，比如令p=0.3，置信度为最小的95%时，当n为20时，p\_value = 1.58 × 10-9因此，在这时直接选择B=S2的数量 > 20且满足置信度95%以上，则p\_value一定小于1.58 × 10-9。

计算完成后将16个分块的输出结果整合在一起。然后建立一个dict包含所有基因的规则，命名为LAW。LAW是一个二维字典，索引是SGDID和状态S，值为包含一系列规则元组(tuple)的列表。形如LAW[SGDID][S] = [(SGDID\_1,S1,SGDID\_2,S2),(SGDID\_3,S3),…]其中，包含规则的元组有四个或者两个的参数，分别代表单输入和双输入的逻辑语句，元组之间的关系为OR。

接下来我们需要构建一个函数，其输入值是包含某时刻表达量的节点字典，即input\_dict[SGDID] = S。根据规则运算后输出一个结构一样的out\_dict，以方便迭代。每个节点通常只满足一类状态变化条件，如果遭遇满足不只一类条件的情况，比如某基因的转换函数符合“1”的条件有10个，符合“2”的条件有1个，符合“0”的条件有100个，则按照10：1：100的比例产生一个随机数来满足节点变化。为了是模拟表达更贴近现实，我们还增加了colony参数。因为通常做Microarray的环境是具有一定压力的，即酵母处于比较激烈的竞争状态中。我们设置了种群数目为10的colony(由于计算能力有限，更大的种群数需要优化程序，或者采用更块的CPU以及分布式计算。)以及1/10的生存率进行选择，更高的生存率可以在输出时取平均值。关于选择函数，我们作出的假设是，具有更平衡的胞内环境的细胞的生存率更高。而细胞内环境的量化指标我们选择正常表达的基因的数量。

3.4 网络可视化

本文的另一个亮点在于将网络间的互相作用用直观的图表示出来。在最原始的尝试中，我们使用基因做为节点，使用基因间相互作用作为边，直接可视化网络。这个方案在实行后立刻被否决，因为我们看到的是一片点，就如图4中的“实心球”一样，事实上，这个“实心球”外的一圈黑色边是节点组成的，而内部的填充色，其实是节点与节点之间的相互作用线填充的。因此将近6000个基因直接可视化，并不现实。于是我们希望只可视化一部分基因，比如度(degree)大于某一个阈值的节点，在我们将阈值提高到40的时候，即当一个基因与至少40个基因相互作用的时候，才被显示出来。终于看到不是一片填充满了的椭圆，而是有空隙的椭圆。但仍然没有实际意义。第三张图是将阈值提升到了70的时候。我们看到了“有意义”的图，在这个图里面，几乎每两个节点都是互相作用的，不过这也并不出乎意料，因为连接数超过70的节点不是调控轴心(hub)就是泛素(ubiquitin)。最后，我们将度的阈值提高到了90，并对显示出来的基因打上了标签(基因的系统命名)。对这些基因进行GO富集后发现确实跟猜想的一样，8个基因(BRE5,UBI4,RPN11,UBP3,RPN10,RPT5,RPN1,SMT3)跟泛素化(deubiquitination)或者泛素相关代谢(ubiquitin-dependent protein catabolic process)有关。P\_value小于10-5。剩下的虽然没有富集，DSN1，CSE4是跟着丝粒以及染色体分裂有关的蛋白，ORC1是跟DNA复制有关的蛋白，GCN5是乙酰基转移酶。(注：由于基因不一定具有标准名standard name，但都有系统命名，因此图例中使用了系统命名作为标签，而文中使用了标准命名是因为标准命名中包含了功能信息。)

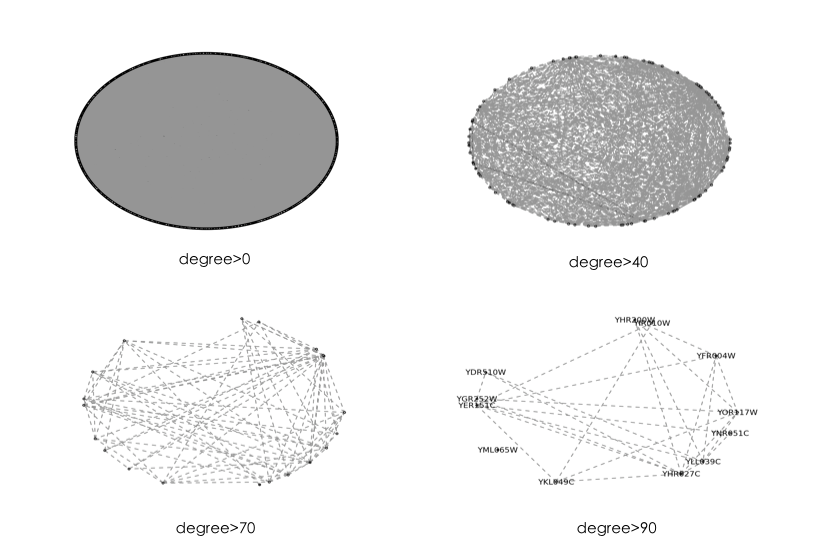


图4：蛋白质互相作用网络直接可视化

第二个尝试中，我们希望能看到更多的基因，我们尝试把网络内部的模块性挖掘出来。我设计了一个动态的算法，使用了先前做蛋白质动力学模拟的经验，给每一个点赋以质量，给整个力场附了一个引力常数，然后用弹簧的模型赋予边，然后使用简单的动力学方程做了一个物理引擎：

R为所有点的坐标，V为所有点的速度，F包含了点与点之间的引力，有边连接的点之间的弹性作用力，以及阻尼。结果表明，对于简单的有模块性的网络，这个引擎的效果很好，如图5显示的是与GAL网络相互作用的所有基因以及运算后的结果。但是该算法对与大规模的网络则无能为力，尤其是模块性并没有那么好的时候。

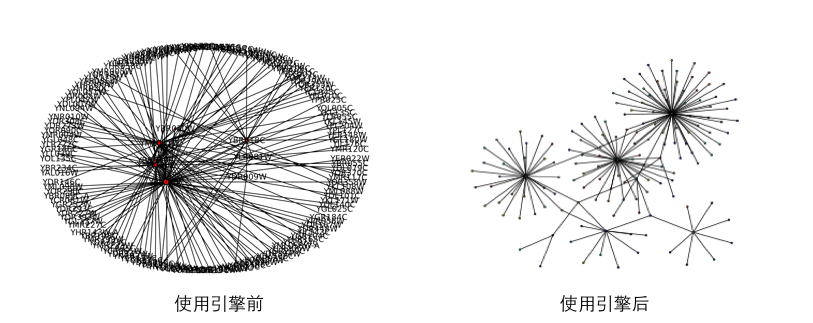


图5：使用物理引擎寻找网络中的模块性

于是我们尝试使用一些其他的算法先计算出模块性，然后根据计算出来的模块性结构用物理引擎进行优化。我们寻找到了一篇kahn等提出的估计模块分块的算法[20]，并使用其算法进行模块性的搜索。下面简述一下算法流程：1）输入网络2）评估每一个点的度，即连接数3）选择度最高的节点，并将直接与其连接的节点所组成的网络提取出来4）切掉该网络中度为1的节点5）剩下的网络节点重复1到5的步骤。使用该算法过程中我们依然遇到了一些问题，比如孤点(被切掉之后的点与剩余网络连接数为0)的处理以及终止条件的判定。此外，最先计算的节点拥有绝对优势的节点数，而剩下的节点连接数虽多，但多数是与第一个节点共享的节点，因此分配下来后形成了更多的孤点。

最后我们的解决方案是使用GO分析的数据，尤其是slim GO的数据，利用人工分析得到的模块性对基因进行分类，然后形成3个网络，分别为Process slim Go，Function slim GO以及Cell Component slim GO。其节点数分别为Process：78，Function：42，Component：24。图6是一个初始化的网络，其中，网络中每一个节点代表一个GO Term，节点的大小(size)为该GO term中包含的基因数目，连接(edge)表示两个GO中有共享的基因，连接的宽度代表两个GO节点之间共享的基因数目。节点大小(size)与基因数目的转换公式以及边宽(width)为：

其中gene\_in\_GO表示该GO下基因的数目，total\_node为整个GO网络的节点总数，G为节点大小转换常数，值为7500(25×300)，即当节点总数为25时，平均节点大小约为300，share\_gene\_num表示该边连接的两个GO节点的共享基因数目。

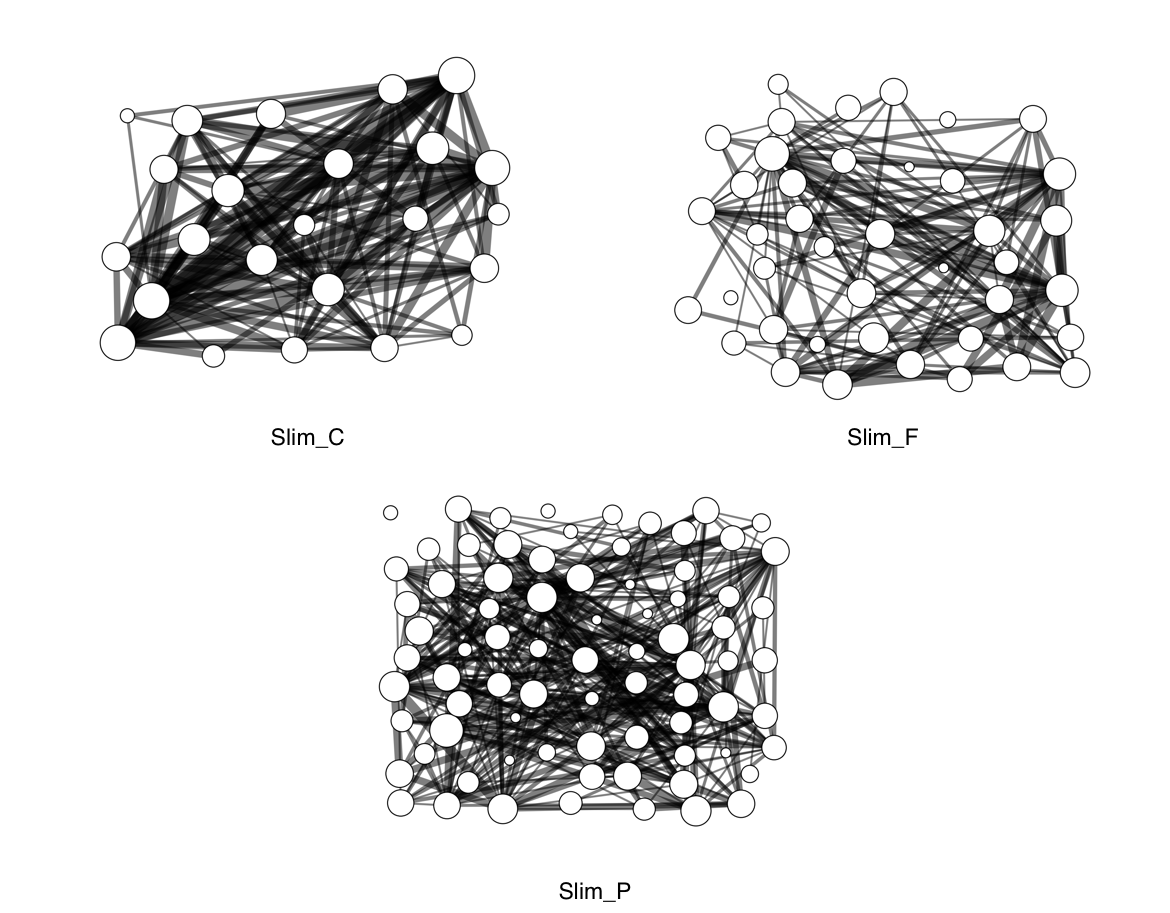


图6：slim\_GO模块初始化

接下来在这个基础上我们对网络的显示进行了一定的优化，优化方案有两种：1）调整节点位置，使GO节点之间的边总长最小2）调整节点位置，使节点边之间的交叉总数最小。在这个优化的过程中，我们使用了遗传算法(Genetic Algorithm)：1）产生一组随机map，即2维坐标的列表 2）产生二十组(population=20)随机的节点顺序，按照顺序赋以坐标值 3）然后计算每一组顺序的分数(fitness)，边的总长度(length)或者边的总交叉数(cross) 4）排序选择分数最高的第三组作为截断值，选择分数等于或高于该截断值的顺序 5）以选择出来的“种群”作为父本，进行交叉互换以及随机突变生成二十组节点顺序 6）重复3-6的过程直到满足收敛条件。其中交叉互换的过程中，由于节点本身的不可重复性，因此不能用通常的交叉互换，我们使用迭代交叉的方法进行。收敛条件为完成10000代优化，或者连续20代没有产生更高fitness的种。该算法被写成了模块供其他python程序调用，详见附录中下载地址。网络的可视化实现使用了Python的NetWorkX模块的网络数据结构[21]以及matplotlib模块的绘图功能[22]。图7是一个对GO:Process网络可视化优化的进程，使用的两种不同的优化函数，可以很明显地看到优化前后之间的差异，优化前后的节点连接完全不变，优化了节点顺序后的显示感觉更加干净。

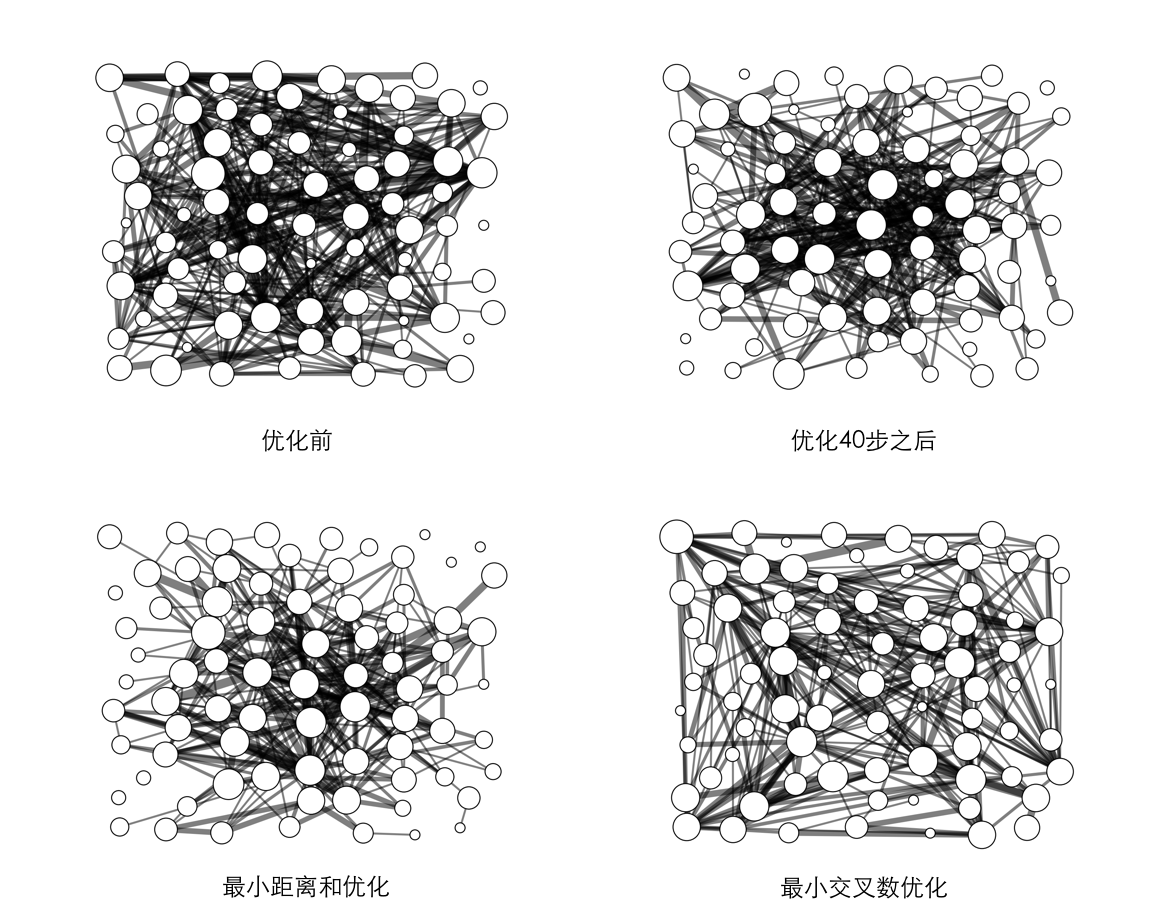


图7：一个优化的例子

接下来，我们希望从网络的角度可视化微阵列的数据，方案是给定一系列的微阵列数据然后根据基因的表达状态（0，1或2）生成三个网络图(sum：所有异常表达节点,1：所有过表达节点,2：所有低表达节点)，网络图“sum”中所有状态非0的基因被统计，每一个GO节点中包含的非0基因总数占GO节点中基因总数的百分比用该节点的颜色的深度来表示，深度越高说明该GO节点表达异常的节点数越多。网络显示的边表示这两个GO节点有共享的基因表达异常，边的透明度越低，表示这两个节点共享的基因表达异常的比率越高，我们的假设是两个节点之间的互相影响一定程度上可以被边的透明度所表示。网络图“1”中则只显示过表达的节点，相对的，网络图“2”中显示低表达的节点。图8显示了表达谱CID=10611304\_0\_0(缺乏氨基酸环境下15min)的三个slim\_GO网络的三种状态，因此一个表达谱可以生成9张网络图。(由于黑白打印的关系，可能无法看出颜色的差别)。而给定一个时序表达谱，根据时序表达谱的状态变化字典，生成节点颜色深度和边动态变化的动画。动画的例子可以在文末附录中的地址下载到。

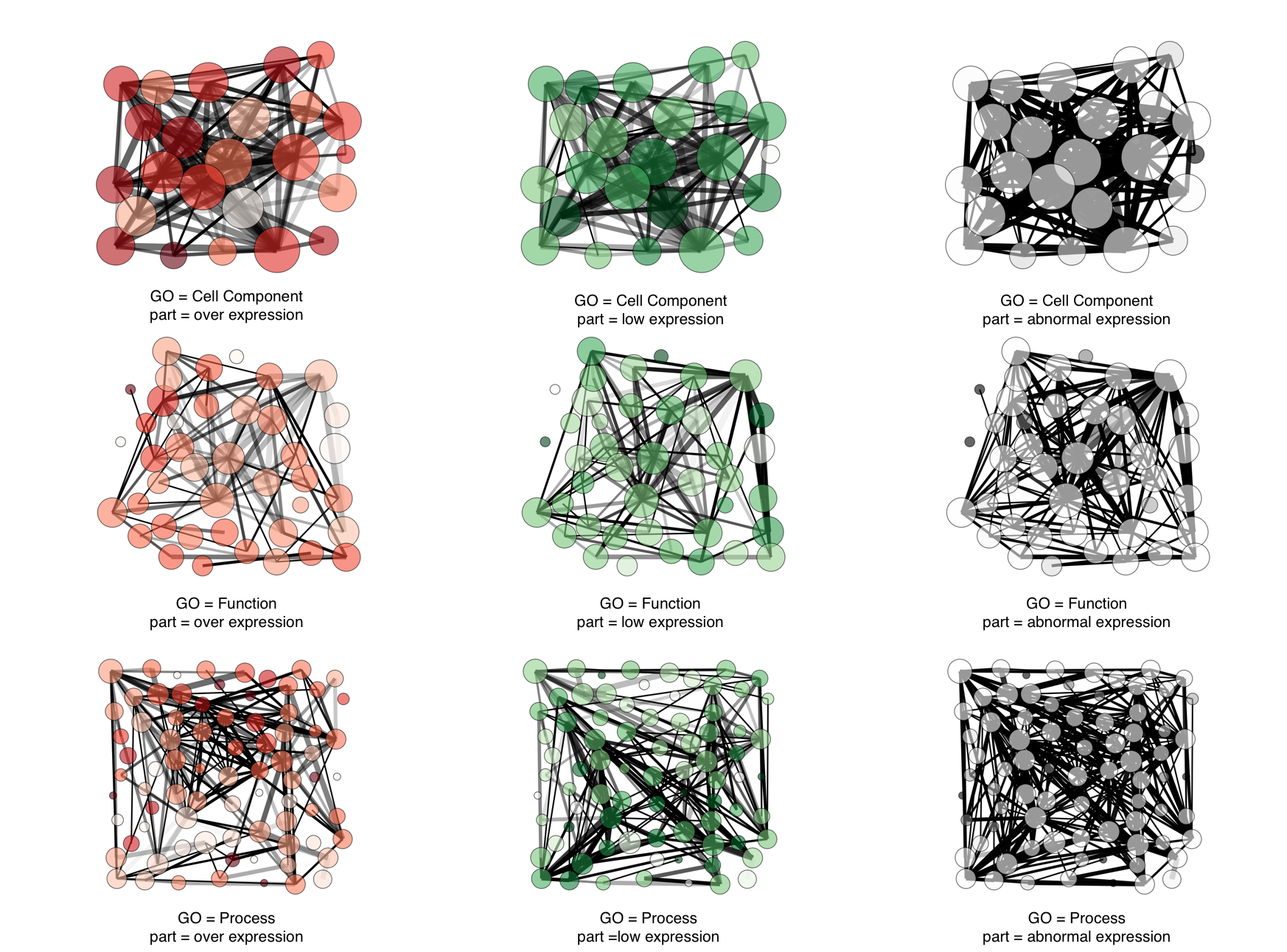


图8：一个CID生成的网络的例子

最后，我们还需要加上标签，因为光看网络的节点变化没有什么意义，我们更希望知道那些节点变化了，因此，我们需要给节点加上标签，由于直接在节点上加描述会令整个图看起来杂乱，因此我给节点本身加上索引，在右侧加上节点名，示例如图9。

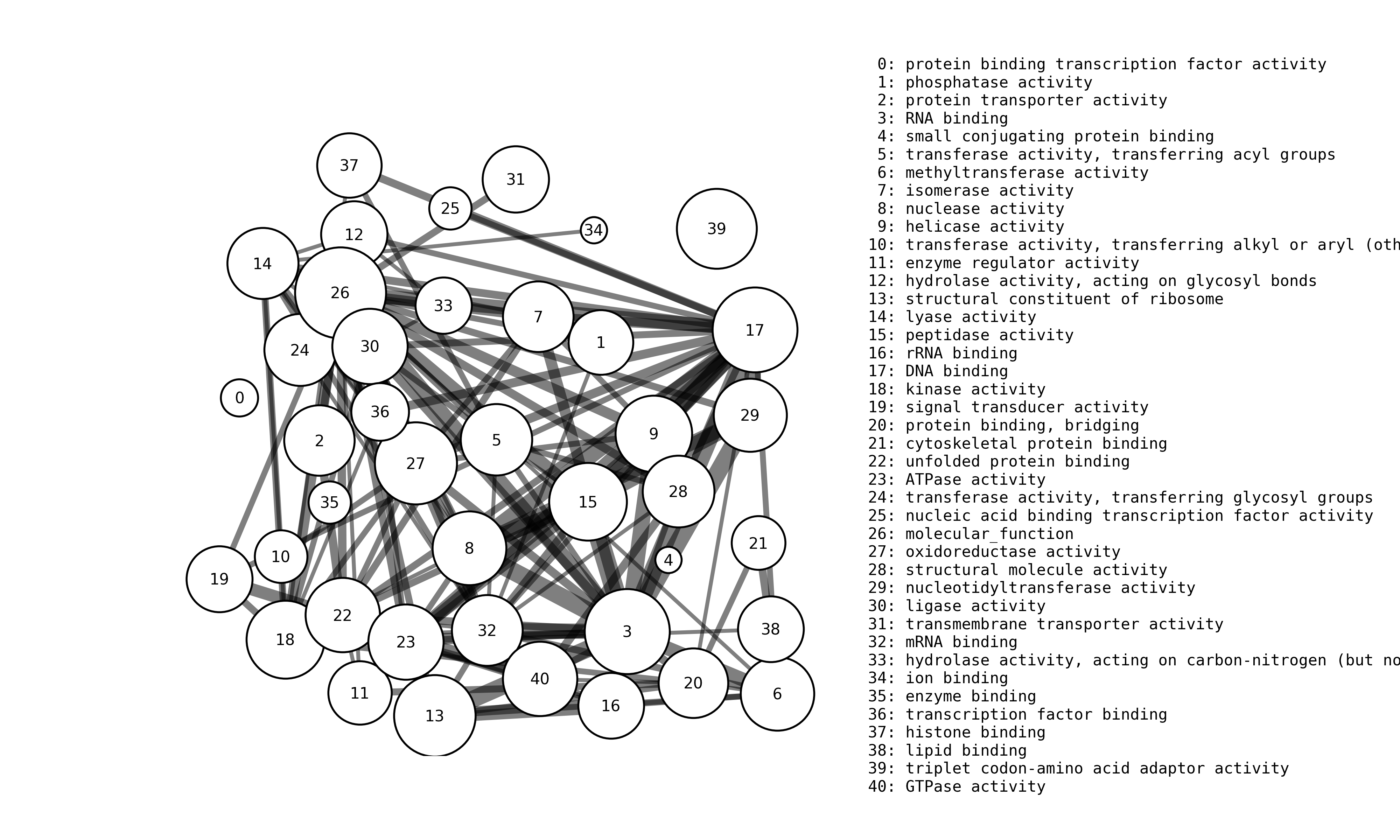


图9：一个加标签后的网络图

3.5 系统验证

系统验证采用交叉验证的方式，即使用一部分数据进行预测和建模，使用另一部分数据进行验证。我们用两种方法进行了验证：1）是输入环境和基因组成预测表达谱 2）是输入初始状态，进行时序迭代，观察后续节点的变化跟时序表达谱的结果的相似程度。

由于我们没有计算环境输入变量的节点。只是筛选出了一部分可能与环境相关的基因，我们假设这些基因只与环境相关，因此这些基因负责环境变量的传递，即只要环境没有变化，该基因的表达状态就不变化。节点的判断是在系统挖掘之后，筛选那些被其他的基因所引用，但是自身没有状态转换函数的节点作为候选。这个假设有一个问题是，即它并不承认环境和别的基因共同作用才产生状态变化的基因。因此在预测时序表达状态的过程中，这一部分基因的状态预测会失败。解决方法是对这些基因使用更大规模的，包含环境参数的系统挖掘，这一步将在以后的深入研究中进行。

第一个验证方法的具体实现是：输入环境相关的节点，约占总结点数的10%。然后根据这些节点的状态，对剩余节点的状态进行估计。验证方法：1）输入已知的环境感知节点状态 2）扫描所有节点，计算由环境感知节点直接作为节点输入函数的节点的状态 3）迭代第二步十次，扫描所有节点，计算环境感知节点以及第二步计算得出的节点的状态作为输入函数的节点状态。4）对2）-3）中计算出的10个状态取概率最大的值，即某个基因出现次数最多的状态为该基因的预测状态。这一步可以抵消一部分因实际调控网络中反应时间不一致的产生的误差。5）比较实际表达谱和预测表达谱。

第二种验证方法是输入一个时序表达谱的初始表达谱数据，预测接下来的表达谱数据。这个验证方案实际上是第一个验证方案的延续。理论上，如果第一个验证方法准确度很高的话，那么即可以凭环境敏感节点预测初始表达谱数据，进而预测时序表达谱的数据。

3.6 系统分析

在系统领域中，系统的鲁棒性和经济性和有效性是三个重要的评价标志，根据系统的目的不同，其侧重的方面也有所不同，比如在发育调控网络中，鲁棒性会比较重要[26]。而比如在大肠杆菌的精氨酸合成系统[27]以及DNA损伤修复系统[28]中，经济性则可能更重要，因此在网络中更可能出现时间程序表达(temporal programs of expression)的单输入模块(Single Input Modular,SIM)，即一个基因调控一系列基因，使之按一定表达顺序表达。从而降低表达成本。有效性和经济性类似，只不过更注重与实际表达效果，因此有效性和经济性放在一起可以并称为“需求规则”。因此我们打算对分析酵母系统中的鲁棒性和经济性。

鲁棒性，即生物在受到外界环境变化时，内环境的稳定性。跟据现成的数据我们得到了一些列时序表达谱，根据这些时序表达谱我们可以推断生物体受外界环境的影响程度和恢复程度。以Gasch Lab的应激性实验[引文]为例，我们将探讨酿酒酵母在热激(Heat Shock)，氧环境(oxidative shocks)，以及缺乏营养物质状态(nitrogen source depletion)。我们的假设是，酵母正常表达的节点占所有节点的比例越高说明酵母受到外环境的影响越小。因此我们从酵母表达谱入手统计酵母的正常表达节点概率和分布。分析流程如下：1）利用已知的数据分析酵母在受到一个持续的不利环境下内稳态的变化曲线以及按GO分类的内稳态变化曲线。利用正常表达节点的百分比表示稳态水平，以时间轴横坐标的曲线图 2）利用上文生成的系统预测时序表达谱，观察内稳态变化曲线 3）利用酵母系统预测单基因删除突变，即低表达锁定状态对整个系统的影响，观察内稳态变化曲线。

需求规则分析，M.A.Savageau定义的基因系统需求规则[引文]是基因的调控模式与环境中基因全表达的概率之间的相关性。但这个假设在调控模式已知并且少量的时候可以使用，对于数百甚至上千的规则的基因几乎无法分析，因此我基于Savageau的需求规则定义了一个更一般化的需求规则，即讨论一个节点的高表达和低表达概率与调控模式的组成的相关性。比如已知一个高表达概率的节点，分析其调控模式中启动调控，抑制调控和恢复调控的比率与这个高表达概率的相关性，并以此说明该节点本身的需求（高表达概率）与供给（启动调控的比率）关系。

此外，在网络领域中还有一个很重要的话题，即模块性，一个模块即一个小系统，包含了输入，系统，输出三个组成部分。通常一个大系统可以被划分为诸多小的模块，每一个模块负责自己的那一部分系统和输入输出。在软件工程领域，模块性甚至是一个设计的基本原则。一个好的系统，模块之间只有输入输出的关系而没有模块内部的影响，即只要模块的输入一致，模块的输出就是一致的。有研究将E.coli的基因互相作用网络跟Linux的系统进程进行了模块性的比较[25]，发现在E.coli的64个模块中有共用节点的模块之间平均共用节点的比率为4.3%，而在Linux的进程访问网络(call graph)的3665个网络中这个比率为80.7%。E.coli的平均节点的重用率为3.5%，即平均每个基因在3.5%的模块中出现，而这一比率在Linux中出现的比率为8.4%。节点在模块的重用率高说明每一个节点的模块性很好，接口一致才能重用。

模块性，是指系统内部模块的独立性。我们定义系统模块性的方法参考了[23,24]的方案，这整个网络的模块性计算公式如下:

其中，C为所有模块的集合，在本文中，即每一个GO节点。E为边(Edge)，E\_c即为该GO节点的边，其中，E\_cin表示该GO节点的内连接数，内连接即该边连接的两个节点都在这个GO分类下，E\_cout表示该GO节点的外连接数，外连接数即该边连接的两个节点有且仅有一个在这个GO分类下。从公式中我们可以发现，网络中如果每个基因作为一个模块，E\_cin为0，E\_cout为该节点的度，整个和为负数，相对的，如果把所有基因作为一个模块，则E\_cin为Etotal，E\_cout为0,公式的求和为0。模块性不会超过1，因此这个模块性的量化公式可以作为评价网络的相对模块性。

我们的预期是，slim\_GO产生的模块性是最好的，而Function，Process，Cell Component三个网络的模块性中Process的模块性最好而Cell Component的模块性最差。因为细胞的进程最复杂，最需要的是模块化和效率化，因此模块性会比较强。而细胞的组成对细胞本身最重要，因此更强调的鲁棒性，而鲁棒性则会削弱系统的模块性，因此细胞的模块性会相对差。

此外，我们还将计算模块性的节点重用率和模块共用节点率。以此与先前的E.coli网络和Linux进程网络相比较。

4. 结果分析：

4.1 数据收集与整合

收集的数据分为两个部分，分别为表达谱数据和酵母已知知识数据。

表达谱数据组成来源与SPELL和SGD的表达谱数据集，数据出处见表1。数据经过1）基因筛选 2）数据整合 3）数据归一化 4）数据布尔化 5）数据存储。生成可以方便存取的数据结构，最终的数据文档形式以2维列表存储，第一行为每一个基因的SGDID，第一列为每一个表达谱的CID，每一个单元格表示这个SGDID在这个CID的表达水平的布尔值，如图10显示了整个文件的一部分(tab分割的文本文件，在纯文本下tab的宽度为4，因此没有对齐，实际上是每个单元格一个数字的表)。

图10：Scer\_Micro.tab部分数据的截图



酵母已知知识来源与酵母基因数据库(SGD)[4]，在本文中，我们下载的源数据均为tab分割的文档。

数据整合的结果为数据库，数据库的结构如图3所示，上文已有陈述。数据库可用SQL语句查询，查询示例以输出结果如图11

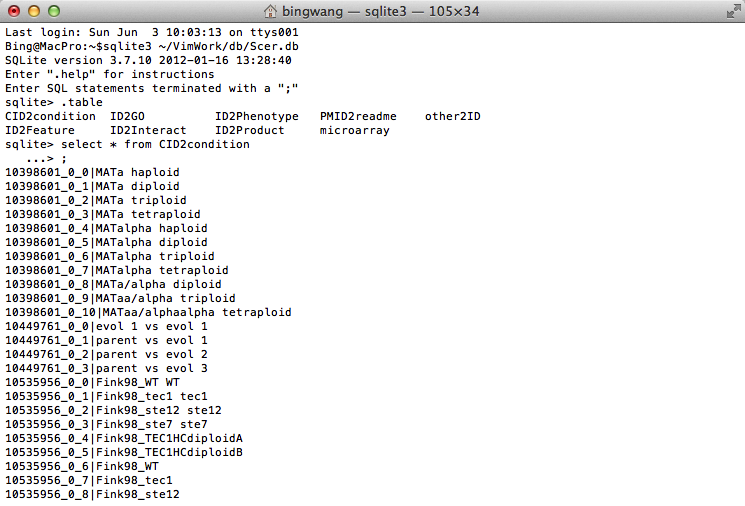


图11：数据库查询范例

|  |  |  |  |
| --- | --- | --- | --- |
| 表1：本文所使用微阵列具体出处 | | | |
| PMID | 作者 | 年份 | 表达谱数目 |
| 18676708 | Abbott | 2008 | 12 |
| 18156292 | Agarwal | 2008 | 6 |
| 15780657 | Aguilera | 2005 | 18 |
| 17158319 | Alper | 2006 | 12 |
| 16896209 | Anderson | 2006 | 26 |
| 19273689 | Anderson | 2009 | 46 |
| 17185230 | Angell | 2006 | 6 |
| 11336698 | Angus\_Hill | 2001 | 8 |
| 16507144 | Aragon | 2006 | 181 |
| 18199684 | Aragon | 2008 | 414 |
| 16543154 | Auld | 2006 | 17 |
| 19155328 | Azzouz | 2009 | 25 |
| 12702356 | Backhus | 2001 | 6 |
| 18378663 | Baerends | 2008 | 8 |
| 11533240 | Baetz | 2001 | 5 |
| 17124610 | Barbara | 2007 | 12 |
| 11752457 | Bedalov | 2001 | 7 |
| 14722110 | Belli | 2004 | 9 |
| 17140446 | Benton | 2006 | 25 |
| 11095743 | Bernstein | 2000 | 18 |
| 18753408 | Berry | 2008 | 77 |
| 15949974 | Boer | 2005 | 12 |
| 17419774 | Boer | 2007 | 15 |
| 19078966 | Borde | 2009 | 21 |
| 19180179 | Bradley | 2009 | 12 |
| 15758028 | Brauer | 2005 | 20 |
| 17959824 | Brauer | 2008 | 36 |
| 15659551 | Brem | 2005 | 131 |
| 12791685 | Bro | 2003 | 7 |
| 18422925 | Brown | 2008 | 6 |
| 14555471 | Bulik | 2003 | 11 |
| 15878181 | Caba | 2005 | 24 |
| 18931682 | Capaldi | 2008 | 85 |
| 11169101 | Carmel-Harel | 2001 | 15 |
| 11675494 | Carroll | 2001 | 12 |
| 17389876 | Carter | 2007 | 90 |
| 11179418 | Causton | 2001 | 45 |
| 18708563 | Chan | 2008 | 10 |
| 18953355 | Chechik | 2008 | 72 |
| 16397762 | Chen | 2006 | 12 |
| 18998772 | Cheung | 2008 | 6 |
| 12419230 | Chitikila | 2002 | 20 |
| 9702192 | Cho | 1998 | 17 |
| 9784122 | Chu | 1998 | 7 |
| 16880382 | Chua | 2006 | 270 |
| 18174152 | Cipollina | 2008 | 12 |
| 18524923 | Cipollina | 2008 | 26 |
| 11988574 | Clark | 2002 | 17 |
| 12006656 | Cohen | 2002 | 11 |
| 15256499 | Cullen | 2004 | 2 |
| 14630934 | Daran-Lapujade | 2004 | 12 |
| 17898166 | Daran-Lapujade | 2007 | 9 |
| 14737171 | De\_Nadal | 2004 | 12 |
| 17933919 | De\_Nicola | 2007 | 12 |
| 9381177 | DeRisi | 1997 | 7 |
| 17538619 | Dettman | 2007 | 8 |
| 15795371 | Dion | 2005 | 55 |
| 17317628 | Dong | 2007 | 10 |
| 16328372 | Dubacq | 2006 | 40 |
| 12820961 | Duvel | 2003 | 30 |
| 11179416 | Epstein | 2001 | 11 |
| 18713793 | Erdogan | 2008 | 6 |
| 16199888 | Eriksson | 2005 | 6 |
| 19005567 | Erlich | 2008 | 12 |
| 10449761 | Ferea | 1999 | 4 |
| 11830665 | Fleming | 2002 | 30 |
| 16542486 | Friedlander | 2006 | 129 |
| 12875747 | Fry | 2003 | 8 |
| 17163986 | Fry | 2006 | 18 |
| 10398601 | Galitski | 1999 | 11 |
| 19234305 | Garcia | 2009 | 18 |
| 15797381 | Gardner | 2005 | 12 |
| 15988024 | Gardner | 2005 | 42 |
| 11102521 | Gasch | 2000 | 138 |
| 11598186 | Gasch | 2001 | 52 |
| 16973600 | Gonzalez | 2006 | 10 |
| 19079573 | Gresham | 2008 | 63 |
| 10922376 | Gross | 2000 | 6 |
| 18762579 | Grund | 2008 | 6 |
| 17166056 | Guan | 2006 | 96 |
| 16122766 | Guo | 2006 | 21 |
| 10611304 | Hardwick | 1999 | 14 |
| 15575969 | Haugen | 2004 | 17 |
| 18281432 | Hazelwood | 2008 | 12 |
| 17785431 | Hickman | 2007 | 6 |
| 19324962 | Holbein | 2009 | 9 |
| 17417638 | Hu | 2007 | 269 |
| 12077337 | Huang | 2002 | 20 |
| 15539461 | Huang | 2004 | 10 |
| 10929718 | Hughes | 2000 | 300 |
| 17407552 | Huisinga | 2007 | 149 |
| 11340206 | Ideker | 2001 | 21 |
| 16407318 | Irvin | 2006 | 14 |
| 17408496 | Iwahashi | 2007 | 15 |
| 18075112 | Iwahashi | 2007 | 9 |
| 17660549 | James | 2007 | 8 |
| 15066785 | Jansen | 2004 | 5 |
| 15870473 | Jansen | 2005 | 6 |
| 15528549 | Jin | 2004 | 6 |
| 17664279 | Jin | 2007 | 20 |
| 16440349 | Johanson | 2006 | 6 |
| 18087042 | Johansson | 2007 | 40 |
| 14570984 | Jones | 2003 | 5 |
| 17999778 | Joseph-Strauss | 2007 | 69 |
| 19029899 | Kao | 2008 | 29 |
| 19023413 | Kaplan | 2008 | 34 |
| 11504737 | Keller | 2001 | 4 |
| 16720269 | Kelly | 2006 | 6 |
| 19456872 | Kim | 2009 | 9 |
| 12269742 | Kitagawa | 2002 | 9 |
| 12854720 | Kitagawa | 2003 | 15 |
| 16377885 | Kitagawa | 2005 | 27 |
| 14734811 | Klevecz | 2004 | 32 |
| 18455824 | Klockow | 2008 | 7 |
| 17241460 | Knijnenburg | 2007 | 30 |
| 19173729 | Knijnenburg | 2009 | 170 |
| 17981122 | Komili | 2007 | 24 |
| 16969341 | Kresnowati | 2006 | 16 |
| 17396017 | Kugou | 2007 | 8 |
| 11154278 | Kuhn | 2001 | 3 |
| 19008939 | Kundaje | 2008 | 24 |
| 16925551 | Kuranda | 2006 | 36 |
| 18927628 | Kvitek | 2008 | 9 |
| 16963631 | Lai | 2006 | 48 |
| 19105839 | Lai | 2008 | 227 |
| 16427747 | Landry | 2006 | 91 |
| 17525304 | Landry | 2007 | 10 |
| 15314654 | Leber | 2004 | 13 |
| 12760044 | Lee | 2000 | 8 |
| 15989963 | Lee | 2005 | 9 |
| 17327914 | Levy | 2007 | 95 |
| 17043222 | Li | 2006 | 48 |
| 17616518 | Liko | 2007 | 8 |
| 10940042 | Lopez | 2000 | 17 |
| 10884426 | Lyons | 2000 | 9 |
| 10535956 | Madhani | 1999 | 11 |
| 16510790 | Malagon | 2006 | 14 |
| 18215224 | Marks | 2008 | 21 |
| 15476558 | Martin | 2004 | 12 |
| 9809554 | Marton | 1998 | 7 |
| 16209719 | Matsumoto | 2005 | 12 |
| 17700863 | Medintz | 2007 | 24 |
| 17890903 | Menacho-Marquez | 2007 | 4 |
| 17601813 | Mendes-Ferreira | 2007 | 33 |
| 16415340 | Mendiratta | 2006 | 9 |
| 12628191 | Meneghini | 2003 | 24 |
| 15192094 | Miyake | 2004 | 4 |
| 14645854 | Mizuguchi | 2004 | 3 |
| 15242642 | Mnaimneh | 2004 | 215 |
| 18424442 | Molina-Navarro | 2008 | 36 |
| 17447102 | Mutiu | 2007 | 9 |
| 17660562 | Mutiu | 2007 | 6 |
| 18508805 | Nag | 2008 | 6 |
| 16429156 | Oberstrass | 2006 | 5 |
| 11102525 | Ogawa | 2000 | 8 |
| 12702272 | Olesen | 2002 | 12 |
| 14623890 | Orlandi | 2004 | 4 |
| 18463633 | Orlando | 2008 | 60 |
| 14595107 | ORourke | 2004 | 133 |
| 18366703 | Pan | 2008 | 6 |
| 16648479 | Parra | 2006 | 20 |
| 17724083 | Parra | 2007 | 12 |
| 17914747 | Pelechano | 2008 | 6 |
| 16246724 | Penheiter | 2005 | 11 |
| 15520001 | Pitkanen | 2004 | 15 |
| 12464633 | Pramila | 2002 | 26 |
| 16912276 | Pramila | 2006 | 50 |
| 11101837 | Primig | 2000 | 24 |
| 14993204 | Prinz | 2004 | 10 |
| 11673473 | Protchenko | 2001 | 4 |
| 18326586 | Protchenko | 2008 | 8 |
| 18522836 | Puig | 2008 | 9 |
| 16923813 | Reinke | 2006 | 17 |
| 18983675 | Ro | 2008 | 14 |
| 10657304 | Roberts | 2000 | 15 |
| 16741729 | Roberts | 2006 | 5 |
| 10811893 | Robertson | 2000 | 12 |
| 14718168 | Rodriguez-Navarro | 2004 | 6 |
| 19426543 | Rodriguez-Quinones | 2009 | 6 |
| 16381818 | Ronen | 2006 | 26 |
| 16023114 | Rosaleny | 2005 | 15 |
| 19711068 | Rossouw | 2009 | 43 |
| 15692568 | Rudra | 2005 | 6 |
| 15456858 | Sabet | 2004 | 18 |
| 15240820 | Saldanha | 2004 | 124 |
| 18533012 | Salusjarvi | 2008 | 15 |
| 15452114 | Sapra | 2004 | 24 |
| 15616569 | Schawalder | 2004 | 24 |
| 16879428 | Scherens | 2006 | 18 |
| 16987817 | Schrader | 2006 | 8 |
| 12740579 | Segal | 2003 | 32 |
| 14668481 | Shakoury-Elizeh | 2004 | 6 |
| 18854817 | Shalem | 2008 | 49 |
| 15371544 | Shapira | 2004 | 70 |
| 18955495 | Shirra | 2008 | 4 |
| 18212068 | Shivaswamy | 2008 | 19 |
| 16870766 | Simons | 2006 | 12 |
| 16332871 | Singh | 2005 | 66 |
| 12135984 | Smith | 2002 | 8 |
| 17551510 | Smith | 2007 | 20 |
| 18416601 | Smith | 2008 | 246 |
| 9843569 | Spellman | 1998 | 55 |
| 19098311 | Sprouse | 2009 | 6 |
| 17453047 | Stern | 2007 | 22 |
| 10725359 | Sudarsanam | 2000 | 12 |
| 15496405 | Tai | 2005 | 24 |
| 15837426 | Takagi | 2005 | 12 |
| 16943074 | Tanaka | 2006 | 6 |
| 17327492 | Thorsen | 2007 | 45 |
| 17179083 | Tompa | 2007 | 8 |
| 17702937 | Torres | 2007 | 62 |
| 10847680 | Travers | 2000 | 10 |
| 16254148 | Tu | 2005 | 36 |
| 17560372 | Urban | 2007 | 19 |
| 15607975 | van\_Attikum | 2004 | 5 |
| 18304306 | van\_den\_Brink | 2008 | 13 |
| 17287356 | Vemuri | 2007 | 6 |
| 17925448 | Venkatasubrahmanyam | 2007 | 16 |
| 11972065 | Wang | 2002 | 58 |
| 12370439 | Williams | 2002 | 8 |
| 18604275 | Willis | 2008 | 24 |
| 18673560 | Wu | 2008 | 4 |
| 10586882 | Wyrick | 1999 | 7 |
| 15647283 | Yamamoto | 2005 | 4 |
| 17158163 | Yarragudi | 2007 | 12 |
| 15998451 | Yeang | 2005 | 7 |
| 18245757 | Yiu | 2008 | 27 |
| 12058033 | Yoshimoto | 2002 | 40 |
| 16461773 | Yu | 2006 | 14 |
| 10744769 | Yun | 2000 | 4 |
| 10894548 | Zhu | 2000 | 26 |
| 18552845 | Zhu | 2008 | 9 |
| 19158363 | Zhu | 2009 | 24 |

4.2 系统构建算法

系统构建的输出结果为逻辑语句集合，分为16个分块进行格点计算，计算结果直接整合到一个文件，大小约为524.6MB。总耗时54小时左右，计算所使用的CPU为8个corei5双核。最后将并行运行得到的结果合并为一个文件。图12是计算结果部分的一个示例：

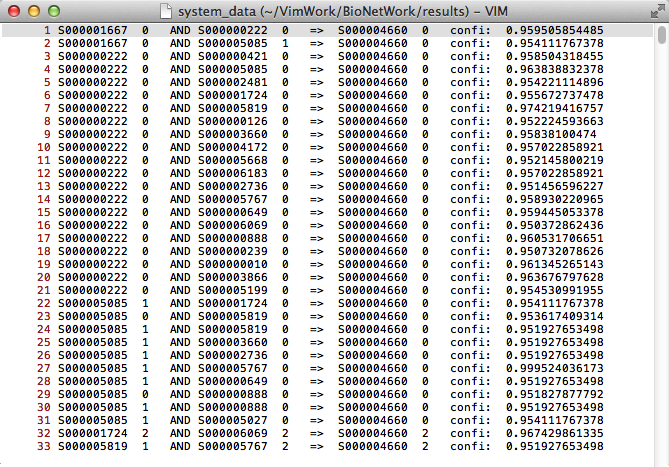


图12：系统文件截图

4.3 网络可视化算法

网络可视化的输出结果为png格式图片，像素为1600×1200。其中优化可视化效果的算法的优化结果如表[3]，（取十次优化进程的平均数）：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表3：网络可视化算法优化结果 | | | | | | |
| GO: | Cell Component | | Process | | Function | |
|  | 距离和 | 交叉数 | 距离和 | 交叉数 | 距离和 | 交叉数 |
| 优化前 | 238.1 | 2091 | 505.3 | 10075 | 220.0 | 2030 |
| 优化后 | 196.9 | 1354 | 267.3 | 2804 | 129.0 | 697 |

图13是网络的可视化进程示例，第一阶段是所有的相互作用被显示在一张图内，无法看清相互作用。第二个阶段是用GO作为基因的集合，并且允许一个基因在多个GO中，边代表共享基因，不显示相互作用。第三个阶段是用遗传算法优化显示出来的图。第四个阶段是根据输入的基因状态，显示GO网络的状态，以及随时间变化的动态。

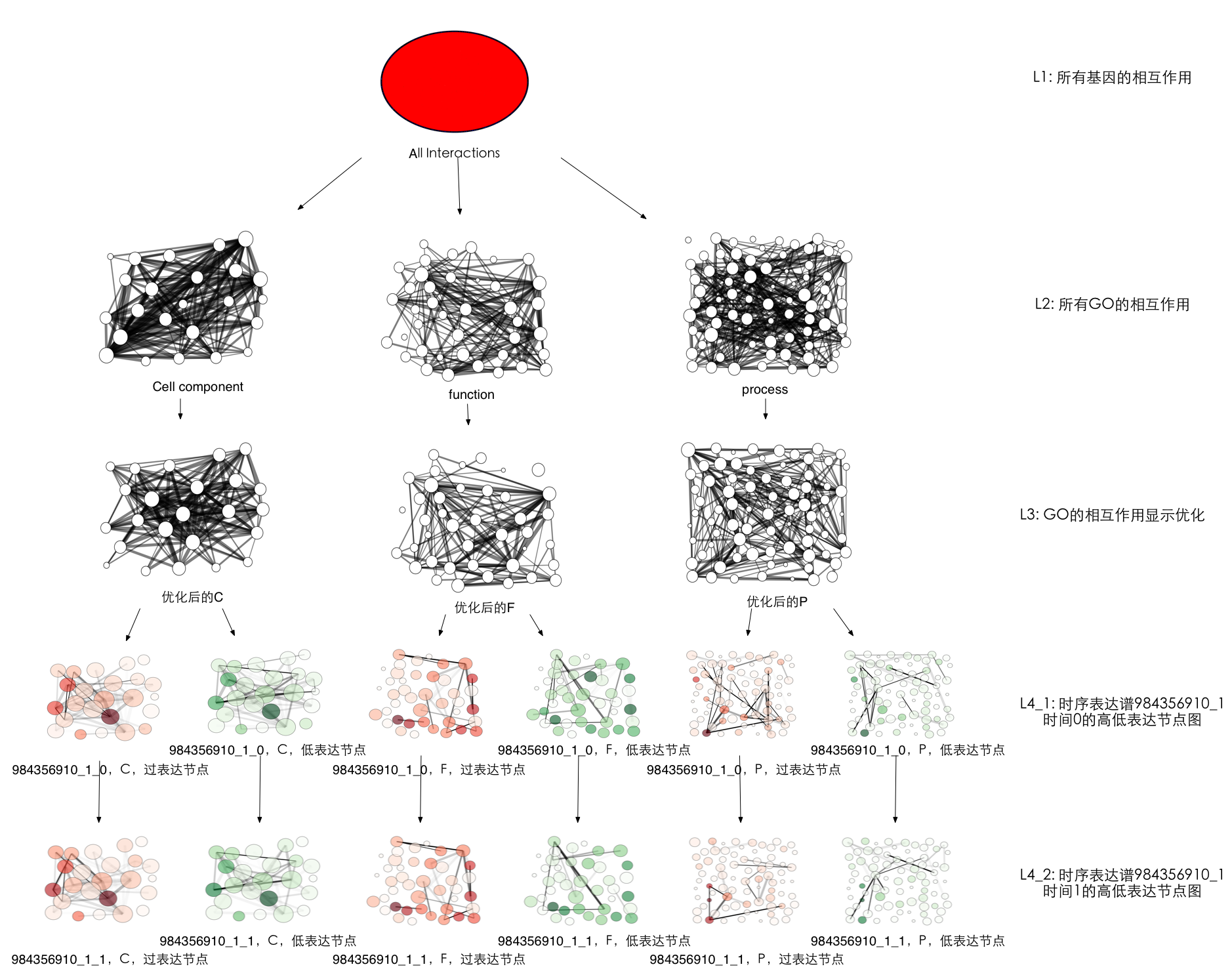


图13：网络可视化进程

4.4 系统组织

这一部分的结果是设计整个系统的输入输出标准，设计如图14。其中带有括号的是数据，括号内的是数据格式，各个模块间的标准输入输出为字典，即一个二维表。酵母系统为逻辑格式，具体见章3.3。酵母基因的其他信息诸如注释和分类则被构建成sqlite3格式的数据。

4.5 系统交叉验证

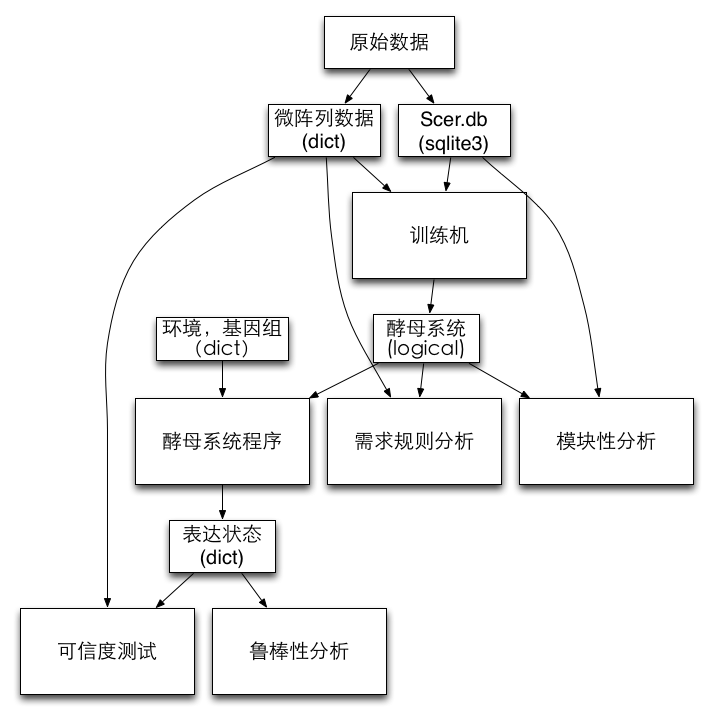


图14：系统输入输出结构

单表达谱预测验证：我们在测试中发现，不同平衡态的表达谱预测出来的准确度差别很大，有高达95%的，也有低至20%的。从某个程度上跟细胞所处的平衡度相关，所谓平衡度是指正常表达基因占所有基因的比例。因此，对整个数据集按照平衡度分为10个等级，分别计算平均预测准确度，结果如表4。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表4：不同平衡态下的预测准确度 | | | | | | | | | | |
| 平衡度 | <10% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | >90% |
| 预测准确度 | 19.6% | 32.5% | 45.0% | 56.1% | 67.5% | 79.4% | 93.5% | 95.1% | 95.7% | 97.0% |

分析：可以发现，当大部分酵母细胞处于比较应激的状态，即不正常表达基因超过总数的70%时，我们的预测效果很差，平均只有45%的基因状态被预测准确，而当这个比例超过90%时，平均只有不到20%的基因能够预测准确。我们认为这是因为在极端的内环境下，细胞的调控本身发生了比较大的改变。即细胞在良好的状态下调控方式都差不多，但在恶劣的内环境下，调控方式差别很大。因此不容易预测，甚至无法预测，因为细胞极有可能会初条件敏感依赖。另一个原因可能是环境特异性的规则没有被挖掘出来，很多需要有环境才能发生的状态被忽略了，从而无法预知一些环境敏感的基因表达状态。

时序表达谱验证：输入初始状态，进行迭代，观察后续状态的相似表达程度直到最终的稳态。本文测试了Gasch论文[引文]里的部分数据，主要是酵母在各种外界环境变化下的应激反应，在这里，外界的环境刺激为2.5 mM DTT，氨基酸和腺嘌呤缺乏环境，氮源缺乏环境，H2O2高氧化环境，29C到33C的热激。结果如表5：

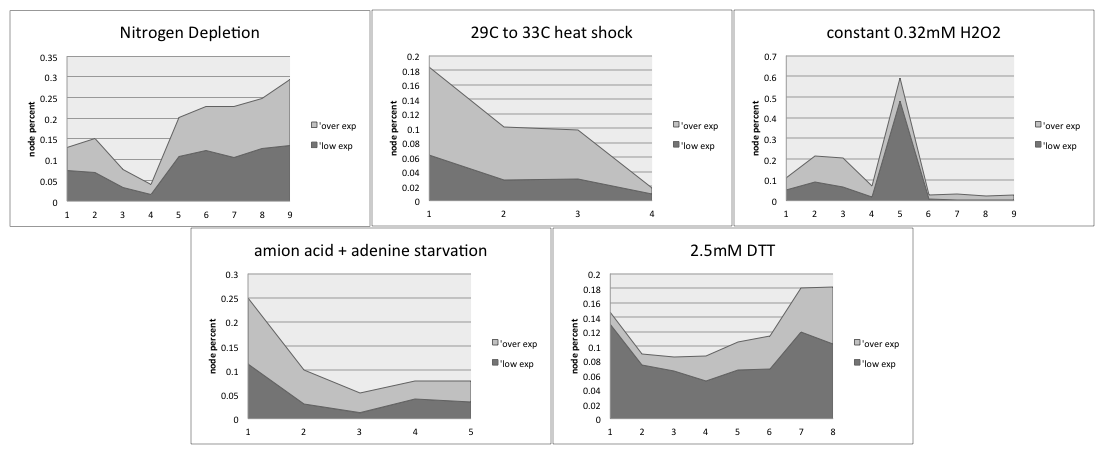
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表5：时序表达谱预测准确度 | | | | | |
| CID | 时长 | 环境相关基因变化率 | 平衡度 | 最高精度 | 发生步数 |
| 2.5mM DTT | | | | | |
| 11102521\_3\_0 | 5 min | 1 | 0.7998 | 1 | 0 |
| 11102521\_3\_1 | 15 min | 0.7389 | 0.8547 | 0.854 | 4 |
| 11102521\_3\_2 | 30 min | 0.745 | 0.8594 | 0.851 | 4 |
| 11102521\_3\_3 | 45 min | 0.7282 | 0.8586 | 0.846 | 4 |
| 11102521\_3\_4 | 60 min | 0.7176 | 0.8397 | 0.829 | 8 |
| 11102521\_3\_5 | 90 min | 0.7038 | 0.8325 | 0.819 | 8 |
| 11102521\_3\_6 | 2 h | 0.6397 | 0.7681 | 0.756 | 12 |
| 11102521\_3\_7 | 3 h | 0.6168 | 0.7676 | 0.753 | 13 |
| Amino acid + adenine starvation | | | | | |
| 11102521\_0\_0 | 0.5 h | 1 | 0.6971 | 1 | 0 |
| 11102521\_0\_1 | 1 h | 0.7368 | 0.8372 | 0.826 | 41 |
| 11102521\_0\_2 | 2 h | 0.6904 | 0.8838 | 0.869 | 41 |
| 11102521\_0\_3 | 4 h | 0.695 | 0.8594 | 0.847 | 41 |
| 11102521\_0\_4 | 6 h | 0.692 | 0.8577 | 0.852 | 41 |
| Constant 0.32 mM H2O2 | | | | | |
| 11102521\_5\_0 | 10 min | 1 | 0.8075 | 1 | 0 |
| 11102521\_5\_1 | 20 min | 0.7692 | 0.7069 | 0.801 | 0 |
| 11102521\_5\_2 | 30 min | 0.7488 | 0.7146 | 0.802 | 0 |
| 11102521\_5\_3 | 50 min | 0.8273 | 0.8427 | 0.890 | 1 |
| 11102521\_5\_4 | 60 min | 0.3925 | 0.3489 | 0.887 | 1 |
| 11102521\_5\_5 | 80 min | 0.8257 | 0.8858 | 0.887 | 2 |
| 11102521\_5\_6 | 100 min | 0.8242 | 0.8801 | 0.884 | 2 |
| 11102521\_5\_7 | 120 min | 0.8273 | 0.8914 | 0.891 | 25 |
| 11102521\_5\_8 | 160 min | 0.8383 | 0.8865 | 0.889 | 25 |
| 29C to 33C heat shock | | | | | |
| 11102521\_7\_0 | 5 min | 1 | 0.7631 | 1 | 0 |
| 11102521\_7\_1 | 15 min | 0.7988 | 0.8426 | 0.836 | 0 |
| 11102521\_7\_2 | 30 min | 0.7665 | 0.8461 | 0.838 | 3 |
| 11102521\_7\_3 | 90 min | 0.7527 | 0.9209 | 0.899 | 44 |
| Nitrogen Depletion | | | | | |
| 11102521\_15\_0 | 30 min | 1 | 0.8185 | 1.0 | 0 |
| 11102521\_15\_1 | 1 h | 0.9008 | 0.799 | 0.918 | 0 |
| 11102521\_15\_2 | 2 h | 0.8779 | 0.8677 | 0.886 | 0 |
| 11102521\_15\_3 | 4 h | 0.8626 | 0.9015 | 0.880 | 8 |
| 11102521\_15\_4 | 8 h | 0.7832 | 0.7495 | 0.837 | 8 |
| 11102521\_15\_5 | 12 h | 0.7176 | 0.7235 | 0.783 | 8 |
| 11102521\_15\_6 | 1 d | 0.713 | 0.7242 | 0.776 | 8 |
| 11102521\_15\_7 | 2 d | 0.6779 | 0.7057 | 0.755 | 8 |
| 11102521\_15\_8 | 3 d | 0.6656 | 0.6624 | 0.719 | 8 |

分析：我们发现，时序表达谱的预测结果并不令人满意，原因可能为1）误差积累，虽然我们选取的系统的单节点规则准确度至少为95%，P\_value的截断点为10-7,但这些误差在时间和庞大的节点数量下可能会被放大许多倍，导致预测结果不佳。2）布尔模型虽然简单有效，但有一个非常大的缺陷，即在一次迭代中，节点之间的反应完全同时进行，因此在布尔模型下，节点是突变的不是渐变的，这个假设令布尔模型无法适应时序表达谱的预测，尽管我们对其进行了一定程度上的改进，但结果仍然不尽如人意，而解决这个问题的方法是使用微分方程(ODE)，但微分方程的缺点则是无法应用于大规模的网络。下一步我们可以尝试，简化微分方程的参数估计条件，结合布尔模型进行预测。 3）环境相关的基因的变化规则未知，因此环境相关的基因的状态差别越大，我们的预测偏差也越大。而就上文分析过的除了环境感知相关的基因外，预测的准确度还取决于细胞内平衡度，因此两者偏差相叠加后导致了结果更大的偏差，解决方案是跟据环境进行分类预测，预测环境相关的基因及其变化规则。

4.5 鲁棒性分析

我们对鲁棒性的定义是在受到外界环境干扰时，保持内稳态的能力。因此我们就使用平衡度来度量稳态的水平，根据时序表达谱分析稳态水平的变化曲线。如图15显示的是酵母在上文5种环境压力下的时序表达谱稳态变化曲线。其中下方的区域是基因组中低表达的基因比例，上方的区域是基因组中过表达的基因比例，因此两者总和越低，细胞内的稳态水平越高，从中我们可以看到三种模式：1）在氮源缺乏(Nitrogen Depletion)和有毒物质(2.5mM DTT)的环境下，细胞会首先呈现一定程度的适应状态，但这个只是暂时的，在短暂的恢复之后会持续恶化，对于这两个环境，我们认为细胞的鲁棒性可能不足以抵挡。2）在氨基酸及腺嘌呤缺乏(Amino acid + adenine starvation)的环境和热激 (29C to 33C)的环境下，细胞会在一开始呈现出异常的状态，随后进行一定程度上的适应，细胞的鲁棒性对这两个环境比较有效。3）强氧化环境(0.32 mM H2O2)，有一个不适应的峰，如果原始数据没有问题的话，那可能是因为此时有一大部分的细胞死亡，剩余的少部分细胞存活下来，细胞的总数减少造成的。

图15：五个环境下稳态变化曲线



4.6 需求规则分析

需求规则是指节点的过表达以及低表达概率与节点规则中指向过表达以及低表达的比例的相关性，结果如表6。表6分为3个字表分别代表节点过表达的情况，节点低表达的情况和节点正常表达的情况。统计每个节点的三种表达情况的概率进行分类统计，规则比例则是系统挖掘过程中挖掘到的指向该表达状态的比例。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表6：基因表达水平与基因规则相关性 | | | | | | | | | | | |
| 节点过表达的情况： | | | | | | | | | | | |
| 概率 | <2% | 2% | 4% | 6% | 8% | 10% | 12% | 14% | 16% | 18% | >20% |
| 节点数 | 97 | 1211 | 1959 | 595 | 267 | 97 | 36 | 23 | 6 | 4 | 0 |
| 过表达规则比例 | 0 | <10-5 | <10-3 | 0.02 | 0.09 | 0.10 | 0.11 | 0.27 | 0.61 | 0.38 | 0 |
| 节点低表达的情况： | | | | | | | | | | | |
| 概率 | <2% | 2% | 4% | 6% | 8% | 10% | 12% | 14% | 16% | 18% | >20% |
| 节点数 | 37 | 326 | 1820 | 1046 | 535 | 331 | 142 | 37 | 15 | 2 | 4 |
| 低表达规则比例 | 0 | 0.01 | 0.06 | 0.13 | 0.25 | 0.23 | 0.29 | 0.41 | 0.50 | 0.83 | 0.83 |
| 节点正常表达的情况： | | | | | | | | | | | |
| 概率 | <80% | 80% | 82% | 84% | 86% | 88% | 90% | 92% | 94% | 96% | >98% |
| 节点数 | 192 | 167 | 257 | 367 | 542 | 1121 | 1333 | 230 | 77 | 9 | 0 |
| 正常表达规则比例 | 0.48 | 0.63 | 0.73 | 0.77 | 0.84 | 0.90 | 0.95 | 0.98 | 0.99 | 1.00 | 0 |

分析：从表中可以明显的看出节点规则与节点表达水平的相关性。简单的说，一个节点的规则中指向某一状态的规则越多，这个节点倾向与这个表达状态的概率越大。我们定义一个基因的需求程度等于表达概率，则该基因的需求程度和调控方式存在着相关性，即一个基因弱倾向与高需求，则该基因更倾向与被激活调控，反之，如果一个基因倾向与低需求，则该基因更倾向于被抑制调控。这一点在大肠杆菌中的单基因调控的基因被Savageau证明[引文]，而在酵母中尚未有人提出。

4.7 模块性分析

在模块性分析中，我们比较了3个GO大类的模块性，节点重用性和共用节点的概率，结果如表7。

|  |  |  |  |
| --- | --- | --- | --- |
| 表7：网络模块性比较 | | | |
|  | 模块性 | 节点在模块中的重用率 | 模块平均共用节点比例 |
| Function | 0.117 | 7.0% | 23.8% |
| Cell component | 0.250 | 28.0% | 19.1% |
| Process | 0.176 | 4.0% | 21.6% |
| S.cer | 0.236 | 7.2% | 10.4% |
| E.coli | / | 3.5% | 4.3% |
| Linux | / | 8.4% | 80.7% |

分析：

5. 结论总结：

5.1 下一步计划

接下来的一步，我们的方向主要有三个，一是提高模型本身的精确度，比如使用隐马尔科夫模型来估计条件输入参数。因为我们观察到的微阵列数据是一个种群的表达平均水平，因此我们观察到的状态变化背后的原因是不明确的，或者说，有很大一部分的特殊的表达被掩盖了。如果说实际细胞是一个马尔科夫状态机的话，那么细胞被观察到的状态是实际状态的一个概率映射，而实际的状态被隐藏起来了，因此使用隐马尔科夫模型来作为系统运算可以更贴近实际的系统，当然，代价是更大的运算量。

另一个是提高系统的训练数据规模和分类精确度，在本论文中，我们由于时间和计算能力的有限，仅选择了1000组微阵列数据，并且，我们并没有将微阵列数据按照环境类型分类，而是统一地不加以区分地捡取数据相关程度，相对的，如果我们能按照时序表达顺序按组训练微阵列数据，我们可以得到时序表达特异的表达规则，甚至可以得到严格的状态转换矩阵，而非之前根据表达相关性得到的状态变化矩阵。另外对环境特异性的分类(比如高浓度酒精环境，低浓度葡萄糖环境，高氧化环境等)让数据根据分类分别训练，可以获得特定环境下的基因表达规则。当然这个分类也可能造成某些环境下的数据不足从而是系统准确度下降。

此外，这个系统并不完善，具体体现在必须要有初始状态的输入，如果没有初始状态的输入，系统将无法迭代到下一步。因此我们的下一步改进是挖掘计算那些只受环境影响和部分受环境影响的节点的表达条件函数。具体方法为：1）计算所有节点的表达条件函数2）寻找那些被当作输入函数，但本身不具备表达条件函数的节点，即被引用但自身不改变的节点3）寻找这些节点异常表达时的环境条件4）计算环境条件与节点表达的相关性5）在节点表达函数中加入环境变量。

5.2 网络应用前景

该网络可以制作酵母数字单基因删除文库(silicon single deletion library)，用于研究单基因缺失对剩余基因组的影响。也可以用来制作双基因删除文库，与单基因删除文库相比较可以预测遗传相互作用(Genetic Interaction)。更大规模的删除文库也可以构建，比如在酵母历史上半乳糖代谢网络丢失事件（Gal Network loss）在S.kud，等三个种中分别出现[引文]。因此我们认为一个网络的丢失只有0或1的状态是个很有意思的现象。数字酵母也许可以解释这个现象并作出一些可验证的预测。而在这个网络丢失之后对剩下的基因组的影响则可以用全Gal network的删除进行预测。

除了制作删除文库以及与之相关的研究外，该网络还可以模拟酵母在不同环境下的适应情况，应用一些修改比如增加几个连接和调控关系，或许能发现有意义的突变进而对真实的酵母进行基因工程改造。应用与工业生产，垃圾代谢，甚至特殊资源勘探等。

致谢：

感谢Chris教授这半年来的悉心指导，您让我认识到了酵母和进化的魅力以及纯粹的科研精神，

周艳红教授，Loewe教授，我的未婚妻邝美华，实验室杨兆万师兄以及师姐江燕华师姐。

附录：

本文涉及到的所有源代码，动画，图例均可以在<https://github.com/BingW/BioNetWork> 下载到。

参考文献：

1. Fraser, H. B., Moses, A. M., & Schadt, E. E. (2010). Evidence for widespread adaptive evolution of gene expression in budding yeast. *Proceedings of the National Academy of Sciences of the United States of America*, *107*(7), 2977-82. doi:10.1073/pnas.0912245107
2. Presgraves, D. C. (2010). The molecular evolutionary basis of species formation. *Nature Reviews Genetics*, *11*, 175-180 ST  - The molecular evolutionary basis of . doi:10.1038/nrg2718
3. Orr, H. A. (2010). The population genetics of beneficial mutations. *Philosophical Transactions of the Royal Society B-Biological Sciences*, *365*, 1195-1201 ST  - The population genetics of benefic. doi:10.1098/rstb.2009.0282
4. Cherry, J. M., Hong, E. L., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E. T., Christie, K. R., et al. (2012). Saccharomyces Genome Database: the genomics resource of budding yeast. *Nucleic acids research*, *40*(Database issue), D700-5. doi:10.1093/nar/gkr1029
5. Hibbs, M. A., Hess, D. C., Myers, C. L., Huttenhower, C., Li, K., & Troyanskaya, O. G. (2007). Exploring the functional landscape of gene expression: directed search of large microarray compendia. *Bioinformatics (Oxford, England)*, *23*(20), 2692-9. doi:10.1093/bioinformatics/btm403
6. Byrne, K. P., & Wolfe, K. H. (2005). The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. *Genome research*, *15*(10), 1456-61. doi:10.1101/gr.3672305
7. de Jong, H. (2002). Modeling and simulation of genetic regulatory systems: a literature review. *Journal of computational biology : a journal of computational molecular cell biology*, *9*(1), 67-103. doi:10.1089/10665270252833208
8. Shmulevich, I., Dougherty, E. R., Kim, S., & Zhang, W. (2002). Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, *18*(2), 261-274. doi:10.1093/bioinformatics/18.2.261
9. Friedman, N., Linial, M., Nachman, I., & Pe’er, D. (2000). Using Bayesian networks to analyze expression data. *Journal of computational biology : a journal of computational molecular cell biology*, *7*(3-4), 601-20. doi:10.1089/106652700750050961
10. Husmeier, D. (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics (Oxford, England)*, *19*(17), 2271-82. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/14630656
11. de Jong, H., & Page, M. (n.d.). Search for steady states of piecewise-linear differential equation models of genetic regulatory networks. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, *5*(2), 208-22. doi:10.1109/TCBB.2007.70254
12. Koch, I., Schüler, M., & Heiner, M. (2011). STEPP - Search Tool for Exploration of Petri net Paths: A New Tool for Petri Net-Based Path Analysis in Biochemical Networks. *Studies in health technology and informatics*, *162*, 113-21. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/21685567
13. Tanay, A., & Shamir, R. (2004). Multilevel modeling and inference of transcription regulation. *Journal of computational biology : a journal of computational molecular cell biology*, *11*(2-3), 357-75. doi:10.1089/1066527041410364
14. Castells-Roca, L., García-Martínez, J., Moreno, J., Herrero, E., Bellí, G., & Pérez-Ortín, J. E. (2011). Heat shock response in yeast involves changes in both transcription rates and mRNA stabilities. *PloS one*, *6*(2), e17272. doi:10.1371/journal.pone.0017272
15. Deplazes, A., Möckli, N., Luke, B., Auerbach, D., & Peter, M. (2009). Yeast Uri1p promotes translation initiation and may provide a link to cotranslational quality control. *The EMBO journal*, *28*(10), 1429-41. doi:10.1038/emboj.2009.98
16. Hu, B., Rappel, W.-J., & Levine, H. (2009). Mechanisms and constraints on yeast MAPK signaling specificity. *Biophysical journal*, *96*(12), 4755-63. doi:10.1016/j.bpj.2009.02.065
17. Venancio, T. M., Balaji, S., Geetha, S., & Aravind, L. (2010). Robustness and evolvability in natural chemical resistance: identification of novel systems properties, biochemical mechanisms and regulatory interactions. *Molecular bioSystems*, *6*(8), 1475-91. doi:10.1039/c002567b
18. Warshel, A., & Levitt, M. (1976). Theoretical studies of enzymic reactions: dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. *Journal of molecular biology*, *103*(2), 227-49. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/985660
19. Gillespie, D. T. (2007). Stochastic simulation of chemical kinetics. *Annual review of physical chemistry*, *58*, 35-55. doi:10.1146/annurev.physchem.58.032806.104637
20. Rhrissorrakrai, K., & Gunsalus, K. C. (2011). MINE: Module Identification in Networks. *BMC bioinformatics*, *12*(1), 192. BioMed Central Ltd. doi:10.1186/1471-2105-12-192
21. Hagberg, A., Swart, P., & S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 11–15.
22. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90-95. A joint publication of the IEEE Computer Society and the American Institute of Physics. doi:10.1109/MCSE.2007.55
23. Leung, I. X. Y., Hui, P., Liò, P., & Crowcroft, J. (2009). Towards real-time community detection in large networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, *79*(6 Pt 2), 066107. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/19658564
24. Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, *69*(2 Pt 2), 026113. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/14995526
25. Yan, K.-K., Fang, G., Bhardwaj, N., Alexander, R. P., & Gerstein, M. (2010). Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks. *Proceedings of the National Academy of Sciences of the United States of America*, *107*(20), 9186-9191. National Academy of Sciences.
26. Eldar, A., Dorfman, R., Weiss, D., Ashe, H., Shilo, B.-Z., & Barkai, N. (2002). Robustness of the BMP morphogen gradient in Drosophila embryonic patterning. *Nature*, *419*(6904), 304-8. doi:10.1038/nature01061
27. Zaslaver, A., Mayo, A. E., Rosenberg, R., Bashkin, P., Sberro, H., Tsalyuk, M., Surette, M. G., et al. (2004). Just-in-time transcription program in metabolic pathways. *Nature genetics*, *36*(5), 486-91. doi:10.1038/ng1348
28. Ronen, M., Rosenberg, R., Shraiman, B. I., & Alon, U. (2002). Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(16), 10555-60. doi:10.1073/pnas.152046799