PPI网络的改进马尔科夫聚类算法

胡庆生 雷秀娟
(陕西师范大学计算机科学学院 西安 710119)

摘 要 蛋白质相互作用(PPI)网络是生物信息学的一个新的研究领域。近年来马尔科夫(MCL)聚类算法在未知蛋白质的功能模块预测方面发挥了重要作用,但是聚类质量不高,为此提出了一种基于突变因子和惩罚因子及重新定义解释聚类结果的MCL聚类算法。该算法采用惩罚因子,惩罚质量较大的吸引子;采用突变因子在算法后期断绝初始转移概率对转移概率的束缚。算法在PPI网络数据集上进行了测试,结果表明该算法不但可以抑制小类的产生,而且聚类结果的质量在Avg. F方面相对于基本MCL算法提高了13.1%。

关键词 MCL聚类算法,惩罚因子,突变因子,PPI网络
中图法分类号 TP391.4,TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.7.023

Improved MCL Clustering Algorithm in PPI Networks

HU Qing-sheng LEI Xiu-juan
(School of Computer Science, Shaanxi Normal University, Xi'an 710119, China)

Abstract Protein-protein interaction (PPI) network is a new research field in the bioinformatics. Recently MCL clustering algorithm has played an important role in the field of predicting the function of unknown proteins. However, the quality of the clustering result is low. An improved MCL clustering algorithm was proposed in this paper, in which the penalty and mutation factors are introduced. New structured regulation operation takes place of expansion operation in the basic MCL algorithm. Experiments were performed on PPI data. And the results show that the new algorithm can not only minimize the number of small clusters, but also improve F-measure and Avg. F value compared with the basic MCL algorithm.

Keywords MCL clustering algorithm, Penalty factor, Mutation factor, PPI network

1 引言

随着人类基因组计划的完成,世界进入到后基因时代,由生物体动态产生并执行遗传程序的蛋白质逐渐进入研究者的视野[1]。研究者发现,生物体的各项生命活动都是由多种蛋白质在时间和空间上协调一致,通过相互作用来完成的。因此,目前的研究对象是PPI网络。对PPI网络聚类分析,可以预测未知蛋白质功能,为疾病治疗、预防和新药开发提供重要的理论基础。

Watts等人于1998年提出了基于人类社会网络的网络模型[2],即小世界网络模型,它既具有与随机网络类似的较短的平均路径长度,又具有与规则网络类似的较高的聚类系数。Barabasi等人于1998年合作进行一项描绘万维网的研究时,发现大多数网络结点(80%)的度很小(不超过4),但极少数结点(不到结点总数的万分之一)却拥有极大的度(超过1000),这种网络称为无尺度网络[3]。PPI网络就是一个具有小世界网络特性的无尺度网络。

传统的PPI网络聚类算法有基于划分的算法[4,5]、基于层次算法[6]以及基于密度的算法[7]。基于划分的算法对孤立点很敏感,基于密度的算法会淘汰掉大量联系比较稀疏的结点。因此产生了一些新颖的算法来分析和研究PPI网络,例如SVM[8]、RVM[9]、MCL[10]以及SVM和RVM的融合算法[11]。

MCL聚类算法属于基于模型的方法,它是van Dongen开发的用于图形聚类的算法,并成功运用到PPI网络聚类分析等领域。该算法基于随机状态转移概率矩阵,不需要预先设定聚类数目,通过概率改变和反复修改矩阵以实现随机流模拟。

MCL聚类算法在生物信息领域得到广泛关注,许多研究者发现该算法对PPI网络聚类非常有效。但是该算法伸缩性差,聚类结果质量不高。Venu Satuluri等人于2009年提出规则马尔科夫(Regularized MCL, R-MCL)算法[12],该算法不仅保留了MCL算法的优点也缓解了其缺点。再将R-MCL算法嵌入到多层结构中衍生出多级规则马尔科夫(Multi-level Regularized MCL, MLR-MCL)算法。相对MCL算法和R-MCL算法,MLR-MCL算法的伸缩性有明显的提高,但是明

到稿日期:2014-07-01 返修日期:2014-10-06 本文受国家自然科学基金青年基金(61100164,61173190),教育部留学回国人员科研启动基金(教外司留[2012]1707号),中央高校基本科研业务费专项资金项目(GK201402035,GK201302025)资助。
胡庆生(1990-),男,硕士生,主要研究方向为生物信息计算,E-mail:huqingsheng321@126.com;雷秀娟(1975-),女,博士,教授,硕士生导师,CCF高级会员,主要研究方向为智能计算与智能优化、生物信息计算等,E-mail:xjlei168@163.com(通信作者)。

显弱于 Metis 算法；MLR-MCL 和 R-MCL 产生更少的聚类个数，从而有效解决了 MCL 聚类结果残片问题。针对 R-MCL 算法的缺点，Venu Satuluri 等人于 2010 年提出可调平衡因子的规则马尔科夫(Regularized MCL With Adjustable Balance，ABR-MCL)算法[13]，再将该算法嵌入到多层结构中，衍生出多级可调平衡因子的规则马尔科夫(Multi-level Regularized MCL With Adjustable Balance，MLABR-MCL)算法。MLABR-MCL 算法在伸缩性及时间复杂度上和 MLR-MCL 相同，但是其聚类结果大类和小类的数量有所减少，提高了聚类结果的 Recall。因为 R-MCL 算法是硬聚类算法，容易把桥结点当作吸引子结点，将两个功能模块合并成一类，所以不能识别较大的重叠模块，不可识别层次模块。为此，Yu-Keng Shih 等人于 2012 年提出软规则马尔科夫('Soft' R-MCL，SR-MCL)算法[14]，该算法的 F-measure 明显高于 R-MCL 算法的值，主要原因是 Recall 明显提高，而 Precision 只提高了一点。针对 SR-MCL 算法较低的 Precision，Yu-Keng Shih 等人于 2013 年提出了考虑负相似度的软规则马尔科夫(SR-MCL-N)算法[15]。该算法的 F-measure 明显高于 SR-MCL 算法，Precision 极大提高，Recall 只是轻微地减小。

ABR-MCL 算法和 R-MCL 算法因为构造的规则化矩阵相似，使得矩阵不收敛并且引入了不必要的公共蛋白质节点(后续有证明)。SR-MCL 算法虽然可以处理桥节点、高重合模块及层次模块，但是算法的时间复杂度提高了(迭代执行 R-MCL)。SR-MCL-N 算法只能针对具有特定金本体结构的 AP-MS 数据。

结合 PPI 网络的数据特性，本文在 R-MCL 聚类算法的基础上，提出了一种改进的基于惩罚因子和突变因子的，并且重新定义解释聚类结果的马尔科夫聚类算法。本文详细叙述了改进的 MCL 算法在 PPI 网络聚类问题上的应用及算法的具体操作步骤。最后在 PPI 网络数据集上进行仿真，并与基本 MCL 和 R-MCL 算法进行比较，结果表明该算法可以弥补基本 MCL 和 R-MCL 算法的缺陷，而且还提高了聚类结果的 Avg. F 和 F-measure。

2 基本 MCL 聚类算法

MCL 算法是迭代交替地在随机矩阵 M 上执行 Expand 和 Inflate 操作，再分别映射到自身。另外，在每次 Inflate 操作之后执行 Prune 操作，直到矩阵 M 收敛。矩阵 M 的定义如下：

$$M(i,j)=\frac{A(i,j)}{\sum_{k=1}^{n}A(k,j)} \tag{1}$$

其中，矩阵 A 是添加自循环的邻接矩阵。

Expand 操作：输入 M，输出 M_{exp}

$$M_{exp}=Expand(M)=M*M \tag{2}$$

Inflate 操作：输入 M 和 Inflate 系数 r，输出 M_{inf}

$$M_{inf}(i,j)=\frac{M(i,j)^r}{\sum_{k=1}^{n}M(k,j)^r} \tag{3}$$

Prune 操作：输入 M，输出 M_{pru}

Prune 操作有 3 种，分别是精确删除(Exact prune)：保留每列前 k 个较大的值，其他的归零；临界值删除(Threshold prune)：取一个临界值 f，大于该临界值 f 的保留，小于该临界值 f 的归零；精确删除和临界值删除的融合：先临界值删除后再精确删除。删除操作之后再对矩阵的列归一化。一般采用临界值删除，计算公式如下：

$$M_{pru}(i,j)=\begin{cases}M(i,j), & M(i,j)\geqslant f\\ 0, & M(i,j)<f\end{cases} \tag{4}$$

其中，f 表示该列的临界值，其定义为：如果 $\max_i(c_i)$ 与 $ctr(c)$ 比较接近，则 f 等于 $a*ctr(c)*(1-b*[\max_i(c_i)-ctr(c)])$，$0<a<1$，$b=1,2,\cdots,8$；如果 $\max_i(c_i)$ 与 $ctr(c)$ 相差较大，则 f 等于 $a*[ctr(c)]^b$，$0<a\leqslant 1\leqslant b$($\max_i(c_i)$ 和 $ctr(c)$ 分别表示结点质量的最大值和中心值)。

MCL 计算之后，得到一个收敛的矩阵 M。解释聚类结果：在矩阵 M 中，第 j 列中只有一个非零值，非零值对应的第 i 行表示结点 v_i 是结点 v_j 的吸引子，即聚类中心，与结点 v_j 类似的结点一起构成同一个类。MCL 算法的伪代码如算法 1 所示。

算法 1 基本 MCL

A:=A+I //每个结点添加自循环

M:=AD^(-1) //构造随机矩阵 M

Do

M:=M_{exp}(M)=Expand(M)

M:=M_{inf}(M)=Inflate(M,r)

M:=Prune(M)

Until M converges

解释聚类结果

3 改进算法的理论

3.1 构造规则化矩阵的理论

p 为某一结点的输出流向量，$q_i(i=1:k)$为该结点的邻接结点的输出流向量，w_i 为该结点到其邻接结点归一化之后的权重，即有 $\sum_{i=1}^{k}w_i=1$。更新该结点的输出流向量，即为其邻接结点与对应权重的线性加权。某一结点与其邻接结点的最小规则化 KL 散度为目标函数，计算公式[12]如下：

$$D(q)=\sum_{i=1}^{k}w_i*KL(q_i\parallel q) \tag{5}$$

式(5)在 $q(x)=\sum_{i=1}^{k}w_iq_i(x)$处取最小值。即可得到：

$$q(x)=\sum_{i=1}^{k}w_iq_i(x)\Leftrightarrow M=M*M_G$$

3.2 引入突变因子的理论基础

R-MCL 更新随机输出流矩阵时采用加权平均，使得聚类过程受初始转移矩阵 M_G 的束缚。在计算矩阵 M 的过程中，M 与 M_G 相乘，增加了 M 中非零值的个数；在 Prune 操作中，较难删除较小的非零值，降低 M 的收敛速度，甚至 M 不收敛。证明如下。

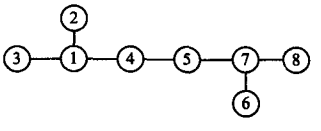


图 1 图 G

反证法证明：图 G(如图 1 所示)的转移矩阵为 M，初始转移矩阵为 M_G。第一步：假设 R-MCL 算法的矩阵 M 能收敛。第二步：按照 R-MCL 算法描述，矩阵 M 收敛之后，再迭代一次算法，所得矩阵 M 一定还是收敛的。第三步：举反例证明第二步的命题是错误的，即可证明第一步的假设是错误的。反例描述如下：

在图1中，转移矩阵为 M，初始转移矩阵为 M_G。

$$M_G=\begin{bmatrix}0.25 & 0.5 & 0.5 & 0.33 & 0 & 0 & 0 & 0\\ 0.25 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0\\ 0.25 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0\\ 0.25 & 0 & 0 & 0.33 & 0.33 & 0 & 0 & 0\\ 0 & 0 & 0 & 0.33 & 0.33 & 0 & 0.25 & 0\\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.25 & 0\\ 0 & 0 & 0 & 0 & 0.33 & 0.5 & 0.25 & 0.5\\ 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0.5\end{bmatrix}$$

假设 R-MCL 算法收敛且求得矩阵 M(由基本 MCL 算法求得)如下：

$$M=\begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

按照 R-MCL 算法描述再迭代一次得到矩阵 M 如下：

$$M=\begin{bmatrix}1 & 1 & 1 & 0.8 & 0.2 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0.2 & 0.8 & 1 & 1 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Prune 操作只能删除列向量中较小的非零值，但是被删除的非零值之和占该列总和的 5%～10%。现在矩阵 M 中的 0.2 显然不能被删除，矩阵 M 不收敛。即可证明第二步中的命题是错误的，从而证明第一步的假设是错误的。

算法初期采用带惩罚因子的新构造的 Regulate 操作代替原始的 Expand 操作，利用惩罚因子缩小各吸引子质量间的差距，减小大类数量及孤立点的个数。因为构造的 Regulate 操作的矩阵和 R-MCL 算法的相似，同样存在与之类似的问题，即矩阵 M 收敛速度慢，引入不必要的公共蛋白质节点，甚至 M 不收敛。为此引入突变的因子，突变因子 m 为突变位置，即为算法迭代次数。算法迭代 m 代之后，采用原始的 Expand 操作，可以有效地加快矩阵 M 的收敛速度，保证矩阵 M 收敛，并且减少不必要的公共蛋白质节点数量。

3.3 重新定义解释聚类结果

之前文献[10]定义的解释聚类结果的方法是：将每行非零值对应的列解释成同一类。假如同一列存在两个或多个非零值，该节点会被解释成多个类的公共节点。因为采用了与 R-MCL 和 ABR-MCL 类似的规则矩阵，并不是严格按照随机流的思想，使得聚类结果出现较多的公共蛋白质节点，甚至类的子类。该解释方法不能识别层次结构模块，而是将其解释成两个类。例如最终计算矩阵如下所示：

$$M_1=\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 0 & 0 & 0 & 0.2\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 1 & 1 & 0.8\\ 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

$$M_2=\begin{bmatrix}1 & 1 & 1 & 0.8 & 0.8 & 0.8\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0.2 & 0.2 & 0.2\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

M_1 解释结果为：1,2,6 为一类，3,4,5,6 为一类。M_2 解释结果为：1,2,3,4,5,6 为一类，4,5,6 为一类。因此，之前的解释聚类结果的方法存在着缺陷。根据 MCL 算法随机流的思想，第 i 列代表第 i 个节点流向其他节点的概率，第 j 行代表其他节点流向第 j 个节点的概率。新定义解释聚类结果的方法是：假如第 k 列第 k 行为零，表示第 k 个节点未能流向自己，而是流向其他节点，那么解释成流向第 k 个节点的节点是第 k 个节点的下一层，在解释聚类结果时将这些节点合并成一类。因此，新定义解释聚类结果的方法可以识别层次聚类模型及类的子类。上例 M_1 和 M_2 的解释结果均为：1,2,3,4,5,6 为一类。

4 改进的 MCL 聚类算法

针对 R-MCL 算法[12]的改进，保留了更新随机输出流矩阵时采用加权平均[12]的特性，构造新的规则化矩阵 M_R。在构造规则化矩阵时引入惩罚因子[13,16]，在算法运行期间引入突变因子[17]。针对 R-MCL 聚类结果非均衡分布且聚类结果质量不高，引入惩罚因子来惩罚质量较大的吸引子，鼓励质量较小的吸引子；针对 R-MCL 中矩阵 M 收敛速度慢，甚至不收敛，引入突变因子来改变在聚类过程中过度受初始转移概率束缚的情况；重新定义解释聚类结果的方法，可以识别层次结构的蛋白质聚类模块及类的子类。

在介绍改进的 MCL 算法之前，先介绍两个定义[13]：

定义 1(结点的质量)　流入该结点的转移概率之和。计算公式：$mass(i)=\sum_j M(i,j)$，其中矩阵 M 为转移概率矩阵。

定义 2(结点的倾向度)　流向其他结点的质量加权。计算公式：$pro(i)=\sum_j M(j,i)*mass(j)$。

R-MCL 在更新随机输出流矩阵时采用加权平均，$Regularize(M)=M*M_G$。吸引子的质量分布直接影响聚类结果，吸引子的质量越大，聚成类的蛋白质数越多。在 R-MCL 中，各吸引子的质量相差较大，直接导致聚类结果中各类的蛋白质数相差较大。为了使聚类结果更均衡，更多的结点聚集到质量较小的吸引子，引入惩罚因子来缩小各吸引子质量间的差距。重新构造规则化矩阵 M_R 代替原来的 M_G。M_R 定义如下：

$$M_R=normalize(M_G*P^{pp}) \tag{6}$$

其中，M_G 是初始转移概率矩阵，P 是由倾向度向量构成的对角矩阵，$pp(0<pp<1)$是引入的惩罚因子。pp 值越小，惩罚

越大，越缩小各邻接结点的倾向度，即缩小各吸引子的质量差别，但是并不是惩罚越大越好，惩罚越大越容易破坏原有的聚类特性。惩罚增大，*Recall* 得到提高，但是 *Precision* 会降低，聚类结果质量会下降。Regulate 操作如算法 2 所示。

算法 2 Regulate 操作

输入：M，M_G，pp

输出：M_R

m=sum(M,2)

$p=M^T * m$

P=diag(p)

$M_R=M_G * P^{pp}$

Normalize(M_R)

改进的 MCL 算法在突变之前迭代交替地执行 Regulate、Inflate 操作，再分别映射到自身，Regulate 操作是引入惩罚因子新构造的操作；在突变之后，算法交替地执行 Expand、Inflate 操作，再分别映射到身身。每次 Inflate 操作之后添加 Prune 操作，直到矩阵 M 收敛。突变位置 m 的获得是由实验摸索取得的。具体实现如算法 3 所示。

算法 3 改进的 MCL

输入：图 G 的邻接矩阵 A，惩罚因子 pp，Inflate 系数 r，引入突变位置 m

输出：聚类结果

A=A+I //构造随机矩阵 M

$M=M_G=AD\wedge(-1)$

Do

If iter<m

M_R=Regulate(M,M_G,pp)

$M=M * M_R$

Else

$M=M * M$

Endif

$M:=M_{inf}(M)=$Inflate(M,r)

M:=Prune(M)

Until M converges

使用新定义解释聚类结果的方法解释聚类结果

5 仿真结果及分析

5.1 实验数据

实验中使用的 PPI 数据集来自 MIPS[18] 数据库，该数据库的部分信息如表 1 所列。以 MIPS 提供的标准结果库为参照标准，来衡量采用本文提出的算法所得到的聚类结果。数据集是以无向带权图的形式表示的。为了计算方便，给每个蛋白质用数字标号，蛋白质间的相互作用权重用一个二维矩阵表示，横坐标和纵坐标所确定的值为该横坐标和纵坐标所表示蛋白质之间的作用强度。

表 1 标准比对数据库信息

聚类数	聚类蛋白质数	总蛋白质数	最大类数	最小类数	类平均数
89	515	1376	35	1	5.76

5.2 评价准则

标准数据库是研究者通过实验和分析得到的正确聚类结果。*Precision*[19] 是聚类正确的结点个数与实验结果中结点个数的比值。*Recall*[19] 是聚类正确的结点个数与标准数据库中结点个数的比值。其计算公式如下：

$$Prec(C_i,C_j)=\frac{|C_i\cap C_j|}{|C_i|} \tag{7}$$

$$Rec(C_i,C_j)=\frac{|C_i\cap C_j|}{|C_j|} \tag{8}$$

其中，聚类结果为 $C=\{C_1,C_2,\cdots,C_i,\cdots,C_k\}$，$C_i$ 是其中任意的一个类，C_j 是标准数据库中的类。

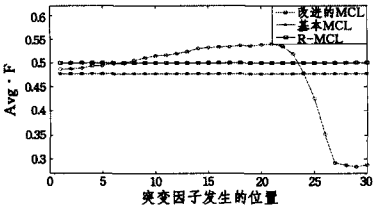
若聚类结果中的类比较大，则 *Recall* 比较高；若聚类结果的类比较小，则 *Precision* 比较高。因此，单纯的以 *Precision* 和 *Recall* 来评价聚类结果不是很合理。每个模块的 *F-measure*（*Precision* 和 *Recall* 的调和平均数）都是独立考虑的，错分的蛋白质无论分到哪个模块都被视为是一样的，这与生物体中各部分组件都有着不同程度的联系是不相符的[20]，而且没有考虑类的大小，因此本文采用调和平均数的加权平均作为聚类结果的评价准则。计算公式如下：

$$F(C_i)=\max_j\frac{2 * Prec * Rec}{Prec+Rec} \tag{9}$$

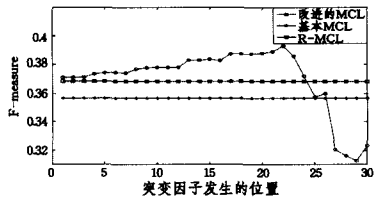
$$Avg.\ F(C)=\frac{\sum_i|C_i| * F(C_i)}{\sum_i|C_i|} \tag{10}$$

5.3 算法性能测试

为了更充分地说明本文所提出算法的优势，将其与基本 MCL 和 R-MCL 进行比较。表 2 是基本 MCL 和 R-MCL 聚类结果的相关数据，表 3 是改进的 MCL 聚类结果的相关数据。图 2 是 3 种算法聚类结果的比较（基本 MCL 和 R-MCL 算法没有突变因子，为了方便比较，在同一图中示出）。



(a) Avg. F



(b) F-measure

图 2 *Avg. F* 和 *F-measure* 的比较

从图 2(a) 中可以看出，改进的 MCL 在突变位置为 21 时，其 *Avg. F* 值最大，比基本 MCL 和 R-MCL 算法分别提高了 13.1% 和 7.8%（改进的 MCL 算法的 *Avg. F* 值为 0.5398，基本 MCL 和 R-MCL 算法的 *Avg. F* 值分别为 0.4771 和 0.5006）；从图 2(b) 可以看出，改进的 MCL 算法在突变位置为 22 时，其 *F-measure* 最大，比基本 MCL 和 R-MCL 算法分别提高了 10.2% 和 6.6%（改进的 MCL 算法的 *F-measure* 值为 0.3928，基本 MCL 和 R-MCL 算法的 *F-measure* 值分别为 0.3566 和 0.3684）。具体数据如表 2、表 3 所列。

表 2　基本 MCL 和 R-MCL 聚类结果的数据

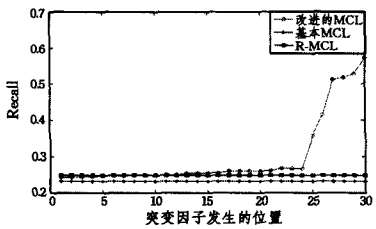
算法	Recall	Precision	F-measure	Avg. F
基本 MCL	0.2307	0.7850	0.3566	0.4771
R-MCL	0.2481	0.6919	0.3684	0.5006

表 3　改进的 MCL 聚类结果的数据

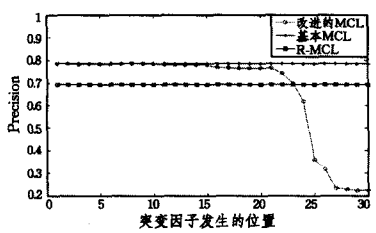
m	Recall	Precision	Avg. F	F-measure
1	0.2428	0.7850	0.4781	0.3709
2	0.2428	0.7850	0.4881	0.3709
3	0.2433	0.7821	0.4889	0.3712
4	0.2450	0.7835	0.4938	0.3733
5	0.2457	0.7837	0.4942	0.3741
6	0.2460	0.7841	0.4987	0.3745
7	0.2456	0.7840	0.4998	0.3741
8	0.2478	0.7866	0.5056	0.3769
9	0.2486	0.7852	0.5104	0.3776
10	0.2490	0.7847	0.5150	0.3781
11	0.2494	0.7808	0.5168	0.3781
12	0.2494	0.7808	0.5168	0.3781
13	0.2535	0.7808	0.5242	0.3828
14	0.2535	0.7808	0.5242	0.3828
15	0.2543	0.7799	0.5324	0.3835
16	0.2547	0.7679	0.5335	0.3825
17	0.2591	0.7667	0.5340	0.3873
18	0.2591	0.7656	0.5361	0.3872
19	0.2591	0.7656	0.5361	0.3872
20	0.2596	0.7652	0.5386	0.3876
21	0.2607	0.7642	0.5398	0.3889
22	0.2669	0.7435	0.5352	0.3928
23	0.2664	0.6974	0.5196	0.3856
24	0.2663	0.6164	0.4792	0.3719
25	0.3566	0.3587	0.4259	0.3576
26	0.4146	0.3178	0.3520	0.3598
27	0.5124	0.2331	0.2920	0.3205
28	0.5172	0.2276	0.2864	0.3161
29	0.5279	0.2220	0.2838	0.3125
30	0.5720	0.2252	0.2872	0.3232

5.4　参数分析

更新随机输出流时采用加权平均，受初始转移矩阵 M_G 的束缚，规则化操作引入惩罚因子来缩小吸引子的质量差距，提高聚类质量。若不引入突变因子，矩阵 M 无法保证收敛。在解释矩阵 M 时，假如某个列向量中存在 n 个非零值，则解释成该结点为 n 个类的公共节点，聚类质量较差，采用新定义的解释聚类结果的方法可以消除一部分公共节点，识别层次聚类模型。突变位置越靠后，这种现象越明显，矩阵 M 未收敛，所以图 3 中的虚线也未收敛，如图 3 所示。



(a)Recall



(b)Precision

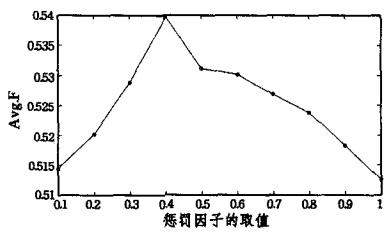
图 3　突变因子对 *Recall* 和 *F-measure* 的影响

突变位置靠前，虽然保证了矩阵 M 收敛，但是未充分利用惩罚因子缩小吸引子的质量差别，*Recall* 增加速率大于 *Precision* 减小速率，聚类质量提高，但是不明显；突变位置靠后，虽然充分利用惩罚因子缩小吸引子的质量差别，但矩阵 M 无法保证收敛，*Recall* 增加速率小于 *Precision* 减小速率，聚类质量下降。所以图 2 中的虚线是先升后降。突变位置的取值就是 *Avg. F* 转折变化对应的位置。

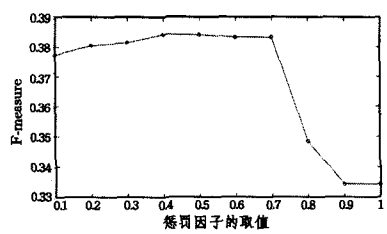
惩罚因子对聚类结果的影响如表 4、图 4 所示。改进的 MCL 算法中引入惩罚因子来缩小吸引子的质量差别，以牺牲 *Precision* 为代价换取 *Recall* 的提高。惩罚因子较小时，吸引子的质量差别较小，对其邻接结点的吸引能力差别较小，*Recall* 增加明显，*Precision* 减少也明显，但两者的变化不是同步的，*Recall* 的增加没能弥补 *Precision* 的减小，聚类质量较差；惩罚因子较大时，吸引子的质量差别较大，对邻接结点的吸引能力差别较大，虽然 *Recall* 提高了，但幅度不大(惩罚因子 pp，$0<pp<1$，其取 1 时，表示没有惩罚)。惩罚因子的取值需要综合考虑 *Recall* 和 *Precision* 的变化，取 *Recall* 弥补 *Precision* 减小之后最大化的点，即 *Avg. F* 最高对应的点。

表 4　惩罚因子对 *Avg. F* 和 *F-measure* 的影响

惩罚因子	Avg. F	F-measure
0.1	0.5143	0.3772
0.2	0.5205	0.3805
0.3	0.5289	0.3816
0.4	0.5397	0.3843
0.5	0.5312	0.3842
0.6	0.5302	0.3834
0.7	0.5269	0.3835
0.8	0.5238	0.3483
0.9	0.5182	0.3343
1	0.5126	0.3343



(a)Avg. F



(b)F-measure

图 4　惩罚因子对 *Avg. F* 和 *F-measure* 的影响

结束语　本文对基本 MCL 聚类算法进行改进，引入惩罚因子和突变因子，重新定义了解释聚类结果的方法，仿真结果表明该方法有效提高了 PPI 网络聚类的质量。但是本文还存在一些问题，算法的速度未能提高，将大类分成许多小类，对类之间桥的划分和类之间重叠部分的划分未能进行有效的解决。这也是许多聚类算法存在的问题，有待于进一步研究。

参 考 文 献

[1]　Von Mering C, Krause R, Snel B, et al. Comparative assessment

of large-scale data sets of protein-protein interactions [J]. Nature, 2002, 417: 399-403

[2] Watts D J, Strogatz S H. Collective dynamics of 'small-world' networks [J]. Nature, 1998, 339: 440-442

[3] Barabasi A L, Oltvai Z N. Network biology: understanding the cell's functional organization [J]. Nature Review Genetics, 2004, 5(2): 101-103

[4] MacQueen J. Some methods for classification and analysis of multivariate observations [C]// Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley, University of California Press, 1967: 281-297

[5] Guo Hai-xiang, Zhu Ke-Jun, Gao Si-wei, et al. An improved genetic K-means algorithm for optimal clustering [C]// Proceedings of 6th IEEE International Conference on Data Mining Workshops. Washington DC: IEEE Computer Society, 2006: 793-797

[6] Zhang T, Ramakrishnan R, Livny M. An efficient data clustering method for very large databases [C]// Proceeding ACM SIGMOD Conference on Management of Data. Montreal, Canada, 1996: 103-114

[7] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in Large Spatial Databases with Noise [C]// Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96). Portland, OR, 1996: 226-231

[8] Ben-Hur A, Horn D, Siegelmann T H, et al. Support vector clustering [J]. Journal of Machine Learning Research, 2002, 2: 125-137

[9] Tipping M E. The Relevance Vector Machine [M]// Advances in Neural Information Processing Systems 12. 2000: 652-658

[10] Dongen S V. Graph Clustering by Flow Simulation [D]. Utrecht: University of Utrecht, 2000

[11] Ruan Pei-ying, Hayashida M, Maruyama O, et al. Prediction of heterotrimeric protein complexes by two-phase learning using neighboring kernels [C]// BMC Bioinformatics. 2014, 15, S2: S6

[12] Satuluri V, Parthasarathy S. Scalable graph clustering using stochastic flows: applications to community discovery [C]// Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA, 2009: 737-746

[13] Satuluri V, Parthasararthy S, Ucar D. Markov clustering of protein interaction networks with improved balance and scalability [C]// Proceeding BCB'10 Proceedings of the 1th ACM International Conference on Bioinformatics and Computational Biology. New York, NY, USA, 2010: 247-256

[14] Shih Y K, Parthasarathy S. Identifying functional modules in interaction networks through overlapping Markov clustering [J]. Bioinformatics, 2012, 28(18): i473-i479

[15] Shih Y K, Parthasarathy S. Identifying protein complexes in AP-MS data with negative evidence via soft Markov clustering [C]// Proceeding BCB'13 Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics. New York, NY, USA, 2013: 440

[16] 岑涌, 罗林开. 一种改善非平衡分布数据 SVM 分类能力的新策略[J]. 计算机与数学工程, 2006, 34(11): 103-105
Cen Yong, Lou Lin-kai. A novel strategy for improving the performance of SVM classification for unbalance distribution data [J]. Computer & Digital Engineering, 2006, 34(11): 103-105

[17] 方霞, 席金菊. 基于变异和启发式选择的蚁群优化算法[J]. 计算机工程与应用, 2013, 49(24): 24-27
Fang Xia, Xi Jin-ju. Ant colony algorithm based on mutation features and selected heuristic[J]. Computer Engineering and Applications, 2013, 49(24): 24-27

[18] Güldener U, Munsterkotter M, Kastenmuller G, et al. CYGD: the Comprehensive Yeast Genome Database [J]. Nucleic Acids Research, 2005, 33(1): 364-368

[19] Zhang Ai-dong. Protein Interaction Networks [M]. UK: Cambridge University Press, 2009

[20] 尤梦丽, 雷秀娟. PPI 网络聚类的评价方法的研究与应用[J]. 计算机科学, 2013, 40(12): 254-258
You Meng-li, Lei Xiu-juan. Study and application of evaluating methods of PPI network clustering[J]. Computer Science, 2013, 40(12): 254-258

(上接第 94 页)

法,通过状态合并等方式压缩模型的状态空间。

参 考 文 献

[1] Colombo C, Pace G J. Recovery within Long-Running Transactions [J]. ACM Computing Surveys, 2013, 45(3): 1-28

[2] Haddad J E, Manouvrier M, Rukoz M. TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition [J]. IEEE Transactions on Services Computing, 2010, 3(1): 73-85

[3] Liu Hai, Zhang Wei-min, Ren Kai-jun, et al. A Novel Selection Approach for Transactional Web Services Composition [C]// 2010 Ninth International Conference on Grid and Cloud Computing. Nanjing, China, 2010: 450-456

[4] Kokash N, Arbab F. Formal Design and Verification of Long-Running Transactions with Extensible Coordination Tools [J]. IEEE Transactions on Services Computing, 2013, 6(2): 186-200

[5] Gaaloul W, Gaaloul K, Sami Bhiri, et al. Log-based transactional workflow mining [J]. Distributed and Parallel Databases, 2009, 25(3): 193-240

[6] Zhang Hua, Wang Qian. Implementing mechanism of service oriented workflow compensation [J]. Journal of Southeast University (Nature Science Edition), 2009, 39(1): 40-46

[7] Zhao Zong-tao, Jun Wei, Li Lin, et al. A concurrency control mechanism for composite service supporting user-defined relaxed atomicity [C]// The 32nd Annual IEEE International Computer Software and Applications Conference. Turku, Finland, 2008: 275-278

[8] Bhiri S, Godart C, Perrin O. Transactional Patterns for Reliable Web Services Compositions [C]// Proceedings of the 6th international conference on Web engineering. California, USA, 2006: 137-144