文章编号: 1000-6788(2010)02-0361-08 中图分类号: TP311.5 文献标志码: A

软件系统网络模体及显著性趋势分析

张 林, 钱冠群, 张 莉

(北京航空航天大学 计算机学院, 北京 100191)

摘 要 借鉴网络模体的研究方法, 对软件网络的局部特性进行了研究. 通过对 138 个开源 JAVA 软件系统的实验比较, 找出了软件系统中 3 种典型的网络模体, 并发现软件系统因规模和交互的不同, 分别与语言网、蛋白质交互网以及信号传输网的局部特性有相似的显著性趋势. 阐述了与实验相关的概念、原理和步骤, 对实验结果进行了分析和讨论, 并给出了进一步的研究方向.

关键词 软件网络; 模体; 显著性剖面; 超家族

Network motif and triad significance profile research on software system

ZHANG Lin, QIAN Guan-qun, ZHANG Li

(School of Compute Science and Engineering, Beihang University, Beijing 100191, China)

Abstract There has been considerable recent interest in network motif for understanding network local features, as well as the growth and evolution mechanisms. In order to discover the local features in software networks, we extended the network motif research methods to software domain. After comparing triad significance profiles from 138 java open source software packages, we found three typical kinds of network motifs. Moreover, the software networks could be divided into 3 clusters which are consistent with the known super-families from various other types of networks. It seems that software scale and interaction may be the reasons causing different motif SP distribution. The concepts,the principles and steps associated with the experiment were elaborated, as well as the results were analyzed and discussed, the direction for further research were given.

Keywords software network; network motif; triad significance profile; super-family

1 引言

大量的实证研究发现, 现实中的许多真实网络, 例如: 社会网、生物网、基因网、神经网、internet 等, 都具有相似的全局统计特性, 如小世界特性 (节点之间的平均距离较短, 同时具有较高的集簇系数)、节点的度分布服从无标度特性[1]. 软件系统可以分解成一系列实体或组成元素, 例如类、子程序、构件等. 将这些元素视为节点, 元素之间的相互关系表示为节点之间的有向边, 软件结构就表现为一个网状的拓扑形态. 因此, 软件系统也被认为是一种人工的复杂网络[2], 我们将这种抽象的软件模型称为软件网络 (Software network). 研究表明, 软件网络同样具有: 1) 小世界特性[3−4]; 2) 度分布表现出无标度特性[4]; 同时, 由于设计优良的软件普遍遵循高内聚低耦合的原则, 所以软件的拓扑结构也具有: 3) 模块化和层次化的特点[5].

复杂系统的研究表明, 具有相似全局特性的网络可能具有完全不同的局部结构, 而这些局部特性对于理解网络的生长和演化机制具有十分重要的意义. 因此, 针对各类网络是否具有共同或相似局部结构的研究愈加得到人们的关注.

收稿日期: 2008-06-18
基金项目: 国家自然科学基金 (60773155); 国家重点基础研究发展计划 (973 计划)(2007CB310803)
作者简介: 张林 (1970–), 女, 博士研究生, 副教授, 主要研究方向软件工程, 复杂网络, 需求工程; 钱冠群 (1978–), 男, 博士研究生, 主要研究方向软件工程, 复杂网络, 软件维护和工程; 张莉 (1968–), 女, 教授, 博士生导师, 主要研究方向软件工程, 过程建模技术, 需求工程.

当前的研究表明, 网络模体 (Network motif) 有助于人们从局部结构上理解复杂网络的生长和演化机制. 网络模体是指在真实网络中出现频率比随机化网络高得多的那些子图[6]. 每个实际网络都可以由其一组特定的网络模体加以刻画, 辨识这些网络模体有助于识别网络中典型的局部连接模式. 同时, 对它们的深入研究, 有助于了解模体本身以及所在网络的特征. 这是因为, 网络模体的频繁出现可能暗示着复杂网络在生长和演化过程中所鼓励的模式, 也可能是网络的演化趋势和进化结果, 对研究网络的形成机理有很重要的意义.

例如: 在有向网络模体中, 三节点的前馈环 (Feedforward loop) 是转录水平调控和神经网络中典型的局部连接模式; 四节点反馈环表示的是电子线路而不是生物系统中的特征模体; 酵母蛋白质交互网络中的模体是其高度进化的反映; 不同物种的转录水平调控网络中, 其模体的类型非常近似, 体现了其进化的趋势.

Milo 等人在网络模体研究的基础上, 对不同领域、不同规模的有向网络进行了分析, 将三元组子图显著性剖面 (Triad significance profile, TSP) 相似的网络放在一起, 称为一个网络超家族 (Superfamily). 他们的研究结果表明, 19 个规模差异很大、功能极其不同的网络大致可分为 4 个网络超家族 (Superfamilies)[7]. 其中, 酵母蛋白质交互网处于一个超家族, 语言网处于另一个超家族, 信号传输网和某些酵母蛋白质交互网处于一个超家族, WWW 网和社会网处于另一个超家族. 这些研究对于认识和理解网络的形成机理, 探讨全局特性和局部特性之间的联系都有重要的意义. 但是, 在他们的研究报告中, 并没有涉及到软件网络.

软件网络模体作为软件网络的基本构成模块, 它的频繁出现, 可能是软件中常用模式的一种体现, 反映了软件潜在的协作和复用方式. 但是, 目前在软件工程领域内, 对软件网络结构特别是网络模体的研究还很少. 对软件网络模体进行深入的研究, 找到软件网络模体的典型特征, 探讨其形成机理对于分析和指导软件系统的演化具有较大的意义. 同时, 通过对软件网络模体的研究, 可以找到与软件网络有相似局部特征的其他领域的网络, 借鉴这些领域的网络结构的演化机理研究, 也可以对软件系统的演化机制有进一步的认识.

本文借鉴 Ron Milo 等人对超家族的研究方法, 深入讨论了以下几个问题: 1) 软件网络中的网络模体特征是什么? 2) 根据局部特性的不同, 软件网络可以被分成几类? 3) 软件网络和哪些网络具有相似的局部特性, 它们之间是否有潜在的联系?

本文的整体组织结构如下: 第 1 节是引言; 第 2 节介绍软件网络的定义; 第 3 节介绍软件网络中网络模体的研究原理及超家族分类方法; 在此基础上, 在第 4 节介绍采用上述方法对软件网络进行的实验设计; 第 5 节对实验中得到的结果进行讨论和分析; 最后, 在第 6 节给出结论, 并指出进一步的工作及研究方向.

2 软件网络的定义

在面向对象的程序设计语言中, 类封装了特定的数据和行为并向外界提供交互的接口. 作为封装和重用的单元, 它隐藏了实现的细节, 并向外界提供服务. 如果将面向对象软件系统中的类/接口看作节点, 类之间的使用/被使用关系或者依赖/被依赖关系看作边, 整个软件系统就表现为一个网状的拓扑形态, 它反映了程序内部复杂的交互和协作关系.

本文集中研究 JAVA 软件系统内部类之间的静态依赖关系. 将软件中的每个类 (Class) 或接口 (Interface) 看作网络中的一个节点, 且不考虑 char, long, double 等基本数据类型. 同时将下列 5 种依赖关系, 映射为节点之间的有向边:

1) 如果类 A 继承子类 B, 那么存在边 A→B;

2) 如果类 A 实现接口 B, 那么存在边 A→B;

3) 如果类 A 包含一个成员变量, 并且类型是 B, 那么存在边 A→B;

4) 如果类 A 包含一个成员函数, 并且函数的某个参数类型是 B, 那么存在边 A→B;

5) 如果类 A 的某个成员函数的实现体中, 包含一个类型为类 B 的变量, 那么存在边 A→B.

图 1 是 Java 软件网络的一个示例 (考虑了 5 种依赖关系).

```
public class Painter extends Person implements Paint-Skill
{
    private Address add;

    public void paint (Furniture furn)
    {
        Logger log;
        // paint furniture and save log info;
    }
}
```

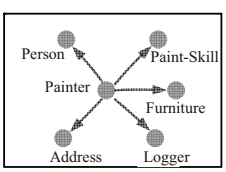


图 1 Java 代码以及对应的有向网络示例

3 网络模体的研究原理

3.1 相关概念和原理

1. 网络模体

尽管网络的形态复杂多变, 但是构成网络的方式和过程却有一定的规律. 近年的研究表明, 网络中各种关系 (节点之间的连接方式) 出现的频率并不是随机的, 有一些是网络的典型连接方式, 即在网络中有些连接方式是反复出现的. 它们在真实系统中出现的次数要远远高于在随机网络中出现的次数. Milo 等人将这种出现频率相对高的连通子图称为网络模体.

不同的网络具有不同的网络模体, 这些 motif 被认为是构造网络的基本 “模块” (Elementary building blocks). 它们被反复的复制和重用, 最终组成大规模的网络.

2. 三节点的连通子图

在有向网络中, 在两个节点之间最多有 2 种关系, 三个节点之间则可以形成 13 种不同的连通关系 (非连通的关系不做考虑). 3 节点的连通子图如图 2 所示. 四节点的连通子图可以形成 199 种关系.

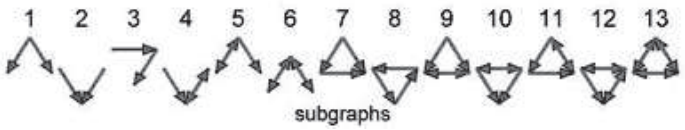


图 2　13 种 3 节点连通子图

出于以下原因, 本文主要针对 TSP 进行软件网络模体的寻找:

1) 在软件网络中, 三节点子图内部的单向和双向边已经可以充分显示出软件单元之间的协作关系.

2) 多数设计模式中静态类图的基本拓扑结构也是 3 节点的连通子图.

3) 避免出现计算上的组合爆炸.

3. 网络模体的检测原理

网络模体的检测原理如图 3 所示, 其中 A 是一个真实网络, B 是与 A 有相同特征的四个随机化的网络[6]. 图 3 中的 A 表示一个真实的网络, 三节点子图 - 前馈环 (Feedforward loop) 用带箭头的虚线表示. 它在图 A 中出现了 5 次. 图 2 中的 B 表示一组 (4 个) 与图 A 有相同特征的随机化网络. 所谓相同特征是指, 随机化网络和真实网络具有相同的节点数, 且每个单个节点都具有相同的入度和出度. 在 B 中, 该子图只是在四次随机化网络中偶然的出现了 2 次. 因此, 可以认为, 该子图就是真实网络中的模体. 这也说明真实网络可能鼓励这种网络模体的出现, 或者可能是该真实网络形成以及演化的潜在机制.

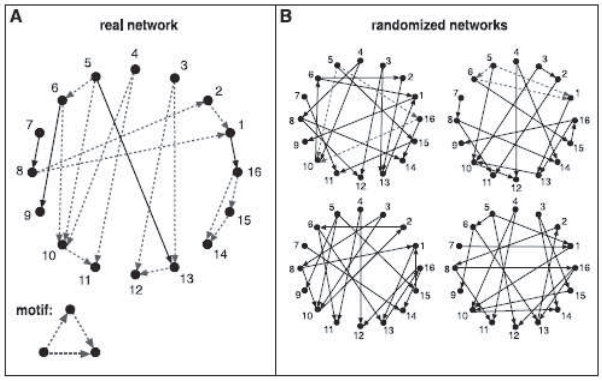


图 3　网络模体检测原理示意图

3.2 子图显著性判断方法

在软件网络中, 哪些连通子图是网络模体的问题, 可以转化为判断哪些连通子图出现的频率相对高. 这需要在真实的网络和具有相同特征的随机化网络之间进行对比.

3.2.1 随机化网络及其构造方法

随机化网络是指和现实网络具有相同的节点数, 相同的出度和入度分布的网络, 它们的边是随机连接的.

一种基本的产生随机化网络的构造方法如下[8]: 在该网络中反复交换随机选取的连接对, 直至整个网络已充分随机化. 例如, 如果某次选取的连接对是 A→ B, C→D, 则把该连接对替换为 A→D, C→B (如果后两个连接中至少有一个已经存在, 那么就不做该次交换). 这样就保证了产生的随机化网络与原始的实际网络具有相同的节点数和出入度.

为保证统计数据的有效性, 随机化网络的数量至少要在 1000 个以上.

3.2.2 显著性 (Significance profile, SP) 判断方法

判断软件网络中子图显著性的方法是: 在真实的网络和具有相同特征的随机化网络之间进行对比, 通过比较子图出现的次数 (规模) 是否有显著性提高, 判断真实网络该连通子图是否是网络模体.

在具体实验中, 为保证子图在随机化网络中出现的随机性, 真实网络中某个子图出现的次数一般要与1000个随机化网络中子图出现的平均次数相比较, 才能判断该子图出现的次数是否存在显著性提高.

该判断方法是一个假设检验的问题.

其零假设是: 真实网络和随机化网络对子图的出现频率没有显著性影响. 备择假设是: 真实网络和随机化网络对子图出现的次数有显著性影响.

为此, 针对每个连通子图 i, 构造以下三个检验统计量: Z_{score}、P 值和 SP 值:

1) Z_{score} 值

$$Zscore = \frac{N_{real} - < N_{rand} >}{std(< N_{rand} >)} \tag{1}$$

$$< N_{rand} >= \frac{\sum_{i=1}^{n} N_{i\_rand}}{n} \tag{2}$$

$$std(< N_{rand} >) = \sqrt{\sum_{i=1}^{n} \frac{(N_{i\_rand} - < N_{rand} >)^2}{n-1}} \tag{3}$$

其中: N_{real} 表示该子图在真实网络中出现的次数; N_{i_rand} 为该子图在第 i 个随机化网络中出现的次数; $< N_{rand} >$ 为 n 次随机化网络中子图出现次数的平均值; $std(N_{rand})$ 是 n 个随机化网络中子图出现次数的标准方差.

2) P 值

P 的定义为

$$P = \frac{\sum_{i=1}^{n} p_i}{n} \tag{4}$$

P_i 定义为: 如果子图在第 i 个随机网络中出现的次数大于或等于在实际网络中出现的次数, 则令 P_i=1; 否则 P_i=0

P 值用来度量样本与零假设之间不一致的程度, 会产生一个 0–1 之间的数值. P 值越小, 零假设不成立的理由越充分.

P 值也称为显著性水平. 当一个假设检验的结果具有较小的 P 值时, 我们就认为这个检验结果具有 “统计显著性”. 一个具有 “统计显著性” 的差异是指, 某个差异与它的标准差相差很大, 具有统计显著性.

3) SP 值

对每个子图对应的 Z_{Score} (记作 Z_i) 作归一化处理就得到对应的 SP_i:

$$SP_i = \frac{Z_i}{\sqrt{\sum Z_i^2}} \tag{5}$$

归一化处理是为了强调子图的相对显著性. 因为大型网络中的模体比小型网络中的模体具有更高的 Z 值, 归一化处理可以避免不同规模的网络对子图出现次数造成的影响. 这对于不同规模的网络之间进行子图显著性比较非常重要[7].

3.2.3 判断依据

某个连通子图是否是网络模体的判断依据:

1) 该子图在与实际网络对应的随机化网络中出现的次数大于它在实际网络中出现的概率是很小的, 通常要求这个概率小于某个阀值 $P(p = 0.01)$. P 值就等于随机样本中, 子图出现次数大于或等于实际网络中子图出现次数的比例.

2) 该子图在实际网络中出现的次数 N_{real} 不小于某个下限 U (如 $U = 4$).

3) 该子图在实际网络中出现的次数 N_{real} 明显高于它在随机化网络中出现的次数 N_{rand}, 一般要求 $N_{real} > 1.1N_{rand}$.

符合上述 3 个条件的连通子图, 即是典型的网络模体.

3.3 网络超家族的聚类分析

在一个真实网络中, 对于每个 3 节点的连通子图都可以得到一个对应的 SP 值, 这组 (13 个) SP 值就反应了真实网络和随机化网络中子图的统计显著性差异.

利用聚类分析原理, 将 SP 取值相似的网络归为一类, 成为一个超家族. 在 Milo 等人做的网络超家族实验中, 根据三节点子图 SP 值的相似程度, 将 19 个不同领域、不同规模的有向网络, 分成了四个超家族 (Superfamily), 如图 4 所示.

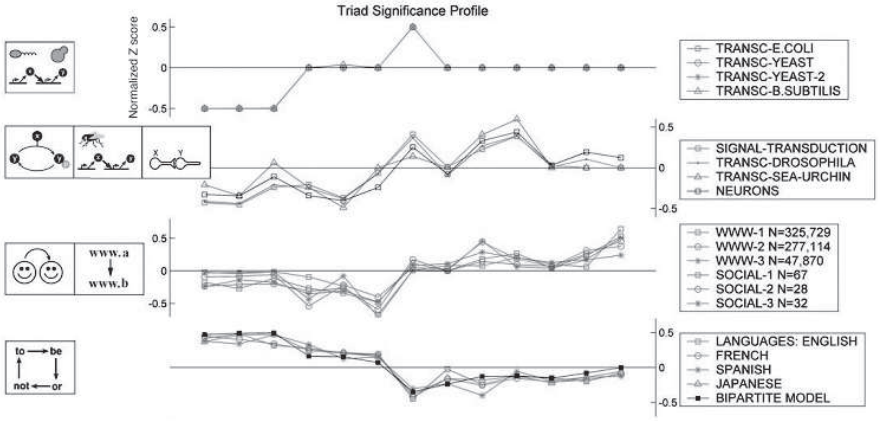


图 4　不同领域网络的 TSP. 有相似 TSP 特征剖面的网络被分到了同一个超家族 [7]

4 软件网络模体显著性实验设计

根据上述原理, 本文对软件网络中的 3 节点连通子图和网络模体进行分析, 并根据 TSP 的相似性对软件网络进行分类. 具体的实验流程如下:

1) 选取样本数据

随机选择来自网上 138 个开源的真实软件系统或软件包作为实验的样本数据. 这些真实软件系统的规模大小不一, 类的数量从十几个 (小型)、几百个 (中型) 到数千个 (大型) 不等.

2) 软件网络模型的建立.

本文采用的模型建立方法如下:

首先, 借助面向 Java 的字节码依赖关系分析工具 DependencyFinder[9], 通过扫描.jar 文件, 获取软件内部各个包、类/接口和特征 (feature) 之间的交互依赖关系;

其次, 将类/接口映射成节点, 类/接口间的交互依赖关系转换为边, 即建立该软件的网络模型. 为让模型能够更加准确地反映软件系统内部的交互和依赖关系, 在模型的建立过程中, 我们仅考虑软件内部的节点以及节点间的依赖关系, 不考虑这些节点对第三方软件包的依赖关系.

3) 判断 3 节点连通子图的是否是网络模体.

使用 Mfinder[10], 计算每个软件网络中 3 节点连通子图的 $Zscore$ 值和 P 值; 符合 3.2.3 节中 3 个判别条件的子图, 即为该网络的网络模体.

4) 软件网络的￥ SP ￥相似性聚类分析以及判别分析

根据公式 1.5, 对每个 $Zscore$ 值进行归一化处理, 得到一组 (13 个)SP 值, 该值反映了各子图的相对显著性.

对 138 个软件网络根据 SP 值相似性进行聚类分析, 并根据前期 Milo 等人的研究数据进行判别分析, 判断软件网络与哪些网络具有相似的局部显著性.

5 实验结果与分析

5.1 软件网络中的网络模体

通过对 138 个软件系统中的 3 节点连通子图进行实验分析, 我们统计得到了 3 个最具显著性的网络模体. 这 3 个网络模体分别是图 2 中的 7 号、10 号和 9 号子图. 具体数据参见表 1.

这 3 个连通子图在所有的软件网络中被认定为是网络模体的比例非常高, 其中 7 号为 96.26%,10 号为 73.83%,9 号为 51.40%. 这 3 种连通子图在真实软件系统中出现的次数要远远高于在随机网络中出现的次数, 说明它们可能是被普遍采用的某些设计模式, 或是被用来执行某些特定的功能或任务. 其中, 7 号连通子图被认为是网络模体的比例最高. 它恰好反映了软件设计中常见的设计方式: 当类 A 与类 B 交互时, 它们通常会同时调用某个类 C 的功能. 通过对设计模式更进一步分析发现, 设计模式中的代理模式就可以被映射成该连通子图的结构. 此外, 其他设计模式也可以被映射成 7 号子图的变体, 如图 5 中的 b, c, d 所示. 从上面的分析可以看出, 之所以该模体在软件网络中出现频率远远高于随机化网络, 是由软件的设计机制造成的.

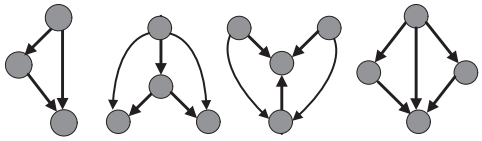
特别地,7 号模体同时也是基因网络、神经网络中的模体. 软件网络和这些网络之间是否存在关联或具有相似的演化机理也值得我们进一步研究.

表 1 3 种典型软件网络模体示例

连通子图	所占比例	真实网络示例 软件名称	节点数	边数	N_{real}	N_{rand} (std)	Z_{score}
7号	96.26%	Emf	89	264	234	133.3	8.96
						−11.2	
		Gef	531	1770	1022	436.6	17.23
						−34	
		Ant	752	3306	3588	2057.5(99.0)	15.46
10号	73.83%	Ant	752	3306	252	130.2	10.33
						−11.8	
		Icu	703	1970	374	83.5	27.08
						−10.7	
		compiler	89	475	57	27.8	5.08
						−5.8	
9号	51.40%	Ant	752	3306	385	108.4	6.84
						−40.5	
		javasvn	378	2250	139	18.9	16.64
						−7.2	
		Compare	378	1365	124	22.9	12.1
						−8.4	

在研究中我们发现, 具有环路拓扑结构的较为复杂的子图 (如 12, 13 号子图), 在显著性统计中几乎没有出现, 这恰好也反映出软件的设计思想. 在软件设计中要尽量避免环路的产生, 否则会增加系统的复杂性和程序的理解难度, 降低软件结构的稳定性.

对于软件系统来说,motif 缺乏较为明确的语义表示, 因此网络模体并不与功能一一对应, 即不能严格地确定功能与结构对应的唯一性. 模体更多地反映了系统结构上的某些限制和特征, 这与 Ingram 等人最新的研究成果是一致的 [11].



(a) (b) (c) (d)

注: 其中 (a) 为 7 号子图, 图 (b) (c) (d) 为 7 号子图自身复制成四节点的子图.

图 5 7 号子图及其被复制之后的四节点子图

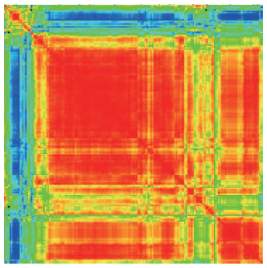


图 6 138 个软件网络 TSP 值相关性系数矩阵

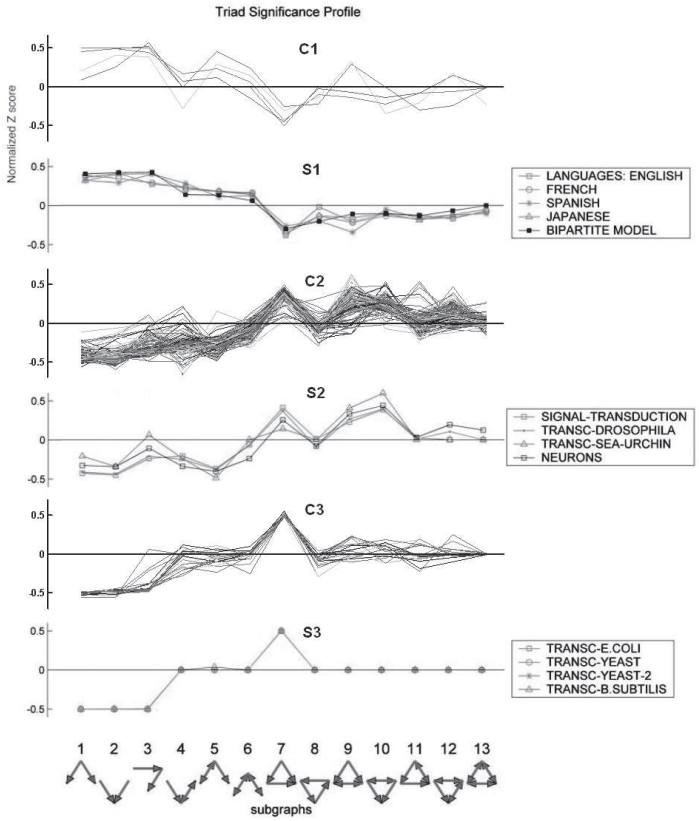
从实验中, 我们也认识到: 不同的软件系统, 它们的网络模体是不同的, 这反映了真实系统在设计和组织上的差异. 模体的拓扑形态与软件的规模以及软件的设计原则有关.

5.2 聚类分析

对 138 个软件网络的 SP 值进行相似性聚类分析, 结果如图 6 所示. 从图 6 中可以看出, 138 个软件网络大致可以分为 3 类. 矩阵左上角是第一类 (C1), 矩阵中心位置是第二类 (C2), 右下角是第三类 (C3). 它们对应的比例大致是 6%, 73% 和 21%.

5.3 显著性趋势分析

根据上述聚类结果, 我们绘制出了这三类软件对应 13 个 TSP 趋势图, 分别对应图 7 中的 C1、C2 和 C3. 同时根据判别分析结果, 可以发现, 它们的趋势分别与 Ron Milo 等人发现的 4 个超家族中的 3 个 (图 7 中的 S1、S2 和 S3) 非常相似. 其中 S1 来自于语言网; S2 来自于信号传输网; S3 来自于蛋白质交互网.



注: 图中 C1、C2 和 C3 是软件网络的 TSP 图. 它们的趋势分别和 Ron Milo 等人研究的语言网 (S1), 信号传输网 (S2), 蛋白质交互网 (S3) 非常相似.

图 7　软件网络 TSP 趋势对比图

从图 7 可以看出, 7 号子图并不是 C1 类网络中的网络模体, 1 号、2 号和 3 号是这类网络的典型网络模体. 而 7 号模体在 C2 网络中是网络模体, 同时 9 号、10 号也是这类网络的模体. 同时, 7 号是 C3 网络的网络模体, 但 10 号和 9 号并不是这类网络的网络模体. 同时 1 号、2 号和 3 号在 C2 和 C3 网络中是反模式. 因此, 可以认为被分成同一类的软件网络不仅具有相同的网络模体, 而且各个模体在网络中的相对重要性也是相似的.

对这三类软件网络 (C1, C2, C3) 的全局特性进行进一步的对比分析, 分析结果如表 2 和图 8 所示.

表 2　3 类 Java 软件依赖网络的基本统计特性

	$< N >$	$< L >$	$< C >$	$\alpha = \left\langle \frac{\log L}{\log N} \right\rangle$
C1	34.2	66.4	0.089	0.764
C2	315.1	1129	0.230	1.204
C3	71	251.7	0.208	1.198

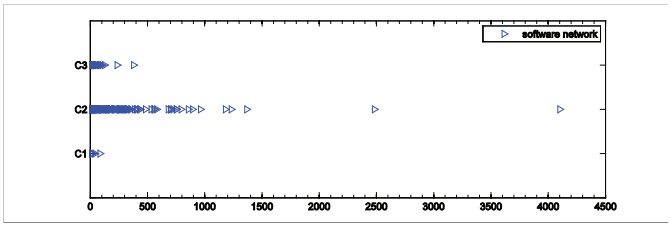


图 8　3 类软件网络的节点数分布

表 2 分别统计了这三类网络的平均节点数 $< N >$、平均边数 $< L >$、平均聚类系数 $< C >$ 以及边数与节点数幂率关系的斜率 α. 图 8 显示了这 3 类网络的节点数的跨度. 从图表中的数据可以看出, 这三类网络可能和软件的规模以及交互关系的密集程度相关.

1) C1 类网络的规模较小, 平均节点数只有 34.2, 平均边数是 66.4, 平均聚系数是 0.089. 这意味着类之间的交互关系比较稀疏, 同时由于这类网络的网络模体是 1 号、2 号和 3 号子图, 因此, 可以知道这类网络的拓扑形态一般是星型或树型结构, 组织的有序化程度较高.

2) C3 类网络的规模中等, 其平均节点数是 71, 平均边数是 251.7, 平均聚类系数是 0.208. 这说明随着网络规模的增加, 软件中类与类之间的交互关系增加迅速, 相对 C1 类网络 $< C1 >$ 和 α 都有大幅度的提高; 同时 7 号模体的出现也意味着软件系统中呈现出了网络化结构.

3) C2 类网络规模更大一些, 其平均节点数是 71, 平均边数是 1129, 平均聚类系数是 0.230, 具有较高交互关系的模体 7 号、10 号和 9 号都是这类网络的典型连接方式, 但 α 并没有特别显著的提高, 这说明在这类软件中, 类与类之间的交互关系更加复杂. 但节点数和边数的增加并没有引起特别大的幂率变化, 这与是软件设计过程中重用和优化的原则有关.

上述的数据仅仅是统计意义上的, 并不意味着某类网络一定具备某种规模. 从图 8 我们可以看出, C2 类 s 网络的节点数跨度很大, 从几十节点到上千的节点都可能存在, 这也进一步证实了在同一个网络超家族中可能包含着规模差异很大、功能极其不同的网络.

对软件网络进行显著性趋势分类, 并和前人的研究成果进行对比, 可以让我们对软件系统及软件网络有更清晰的认识, 同时也可以借鉴和利用其他领域的研究成果, 对软件复杂网络的形成机理及演化趋势进行更深入的认识提供参考.

6 结论和进一步的工作

软件网络已经被证实具有复杂网络的共同特性, 如小世界特性, 无标度特性; 同时由于软件设计所遵循的高内聚低耦合原则, 因此软件内部还具有模块性和层次性, 从复杂网络的角度, 对软件网络进行的研究已经逐渐引起专家和学者的广泛兴趣. 但是, 相比其他领域 (如生物网、社会网、WWW 网), 对软件网络的研究才刚刚开始, 特别是在局部特性的研究上还显得较为薄弱. 本文借鉴 Milo 等人在网络模体以及超家族分类方面的研究方法, 对 138 个开源 JAVA 软件进行了网络化建模, 并进行了网络模体分析, 找出了较为典型的 3 种模体, 并对其出现的原因进行了分析. 在此基础上, 对这些软件网络进行了子图显著性趋势 (SP) 分析以及相似性分析, 发现软件网络就此可以被分成三类, 这三类软件网络与 Milo 等人的研究成果中的语言网、信号传输网以及蛋白质网的趋势相似, 并对软件网络的分类进行了进一步的讨论.

这些研究使我们对软件网络有了进一步的认识, 使我们可能从同一超家族中其它网路的研究方法和研究成果中获益, 进而更加深入的理解软件网络的生长和演化机制. 同时也使我们进一步思考, 导致三类软件网络内部 SP 趋势相似性的原因是什么? 造成软件网络之间子图显著性趋势差异的原因是什么? 这种显著性差异的相似性是否与软件的适应性、编程语言、设计模式和软件规模有关? 此外, 软件网络中全局特性和局部特性之间存在着什么关联? 是否能够彼此互为预测? 这些都值得我们进一步深入研究和探讨.

参考文献

[1] Newman M E. The structure and function of complex networks[J]. SIAM Review, 2003, 45(2): 167–256.
[2] Myers C R. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs[J]. Physical Review E, 2003, 68(4): 46116.
[3] De M A, Lai Y C, Motter A E. Signatures of small-world and scale-free properties in large computer programs[J]. Physical Review E, 2003, 68(1): 17102.
[4] Valverde S, Cancho R F, Sole R V. Scale-free networks from optimal design[J]. Europhysics Letters (EPL), 2002, 60(4): 512–517.
[5] Hyland W D, Carrington D, Kaplan S. Scale-free nature of java software package, class and method collaboration graphs[R]. MIND Laboratory,University of Maryland College Park, 2006.
[6] Milo R, Shen-Orr S, Itzkovitz S, et al. Network motifs: Simple building blocks of complex networks[J]. Science, 2002, 298(5594): 824–827.
[7] Milo R, Itzkovitz S, Kashtan N, et al. Superfamilies of evolved and designed networks[J]. Science, 2004, 303(5663): 1538–1542.
[8] Kannan R, Tetali P, Vempala S. Simple Markov-chain algorithms for generating bipartite graphs and tournaments[C]. New Orleans, LA, USA: ACM, 1997.
[9] DependencyFinder[DB/CD]. 1.2.0 ed. http://depfind.sourceforge.net/, 2007.
[10] Mfinder[DB/CD]. 1.2 ed. http://www.weizmann.ac.il/mcb/UriAlon/groupNetworkMotifSW.html, 2008.
[11] Ingram P J, Stumpf M, Stark J. Network motifs: Structure does not determine function[J]. BMC Genomics, 2006, 7(1): 108.