基于复杂网络的二阶段软件聚类方法

钱冠群 张 林 张 莉

(北京航空航天大学 计算机学院，北京 100191)

摘 要：将复杂网络社区检测中的GN(Girvan-Newman)算法引入到软件聚类中，针对GN算法中存在的计算量大、可能产生小规模社区的缺陷，提出了一种二阶段聚类方法. 首先基于结构模式对软件网络进行聚类. 通过识别和聚类软件网络中3种常见的结构模式：卫星结构、链结构和拓扑相似结构，可以有效地减小网络规模. 其次，在限制模块大小的前提下利用改进的GN算法进行聚类. 如果介数最大边的删除会导致生成的社区规模小于预定值，那么放弃删除该边，转而尝试介数次大的边. 实验结果表明：二阶段聚类算法可以有效地改善软件聚类效果，提高现有社区划分算法在大规模软件中的适用性.

关 键 词：遗留系统；逆向工程；再工程

中图分类号：TP 311

文献标识码：A **文 章 编 号**：1001-5965(2009)12-1438-05

Two-phase software clustering method based on complex network theory

Qian Guanqun Zhang Lin Zhang Li

(School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract: GN(Girvan-Newman) algorithm, a famous community detection algorithm, is introduced into software clustering. In order to overtake the weakness of high computation complexity and avoid generating small scale modules, a two-phase software clustering method is proposed. Firstly, cluster software based on its structure pattern. 3 structure patterns are identified, including: star structure, link structure and topology similarity structure. Cluster these structure patterns could efficiently reduce the scale of software network. Secondly, use modified GN algorithm to cluster software. If the remove of the edge with maximal betweenness would produce a module whose scale is smaller than the value set in advance, this remove action is forbidden. The edge with secondly maximal betweenness is tried. The experiment results show that the two-phase clustering algorithms can improve the effect of software clustering and be applied in the large-scale software.

Key words: legacy system; reverse engineering; reengineering

软件聚类技术提出的目的是通过从源代码中提取出软件中的子系统和它们之间的关系，提高人们理解、分析和改造遗留系统的效率. 现有的自动聚类方法中，较具代表性的有：ACDC方法[1]、Bunch方法[2]、LIMBO方法[3]等. 但这些方法还不能满足大型软件的维护需求，需要进一步改进和提高[4]. 另一方面，随着软件规模的不断增长，软件俨然成为一种人工的复杂系统. 于是，人们开始从复杂系统的角度，借助复杂系统的相关理论，特别是复杂网络理论，来分析和研究软件系统，希望为软件的设计、开发和维护提供理论依据[5].

现有的研究表明，软件可以用复杂网络加以表示[6]，而真实世界中的复杂网络，大都具有相似的抱团特性. 整个网络可以划分成若干个社区，社区内的节点连接紧密，而社区之间节点的连接相对比较稀疏[7]. 通过分析网络的社区结构，可以推断社区中节点的性质，进而理解网络整体的结构和功能特性. 这与软件聚类的目标不谋而合.

收稿日期：2008-11-20

基金项目：国家自然科学基金资助项目(60773155)；国家973基金资助项目(2007CB310803)

作者简介：钱冠群(1978—)，男，浙江宁波人，博士生，qianguanqun@cse.buaa.edu.cn.

本文将文献[7]提出的 GN(Girvan-Newman)社区划分算法引入到软件聚类中,针对 GN 算法应用到软件聚类时存在的计算量大、可能产生小规模社区的缺陷,结合软件自身的结构特点,提出一种二阶段软件聚类方法.

1　GN 算法简介

GN 算法使用边的介数来衡量边在社区间连通中的重要程度.所谓边介数,就是这条边在网络的节点间最短路径中出现的次数.通过不断从网络中删除介数最大的边,使网络分裂为若干个社区[7].同时,为了衡量不同网络划分的质量,文献[8]还引入了模块度的概念:

$$Q=\sum_{i=1}^{k}\left[\frac{l_i}{L}-\left(\frac{d_i}{2L}\right)^2\right] \tag{1}$$

其中,l_i 为模块 i 内部边的条数;d_i 为模块 i 中节点的度的总和;L 为网络中所有边的条数.模块度越大,表示划分的效果越好.

2　二阶段软件聚类方法

GN 算法在社会学、生物学等领域得到广泛的应用,但应用到软件聚类中,还存在一些缺陷:

1) GN 算法的复杂度较高,不能适用于大型软件的聚类.GN 算法的复杂度是 $O(L^2N)$,N 为网络的节点个数.若将 GN 算法应用到大型软件的聚类,有必要根据软件网络的特点,减小网络规模.

为此,本文提出 3 种不同的软件网络结构模式:卫星结构、链结构和拓扑相似结构.通过聚类上述 3 种结构模式来简化网络.

2) 软件聚类的结果应当避免聚类过大或者过小[4],划分的模块太多或者太少,都不利于开发人员从高层上对系统进行理解.但是,GN 算法的聚类结果中可能出现过小的社区,甚至是单节点社区.

GN 算法采用迭代的方式删除介数最高的边,而且一旦分裂以后就无法再合并,因此有可能无法达到模块度的全局最优解,甚至是局部最优解.特别是在算法执行中出现友邻节点个数为 1 的节点时,该节点所连接的边可能具有较高的介数,从而被优先分裂成为单节点社区.但是,根据式(1)中模块度的定义,若将单节点社区和与它相邻的社区合并成一个社区,模块度可能会上升.

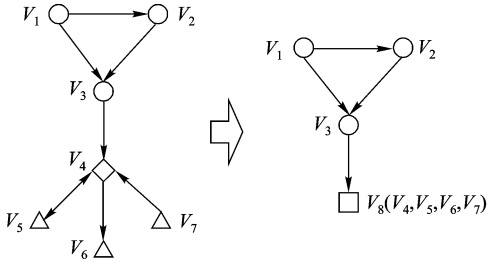
因此,有必要在 GN 算法中增加对社区规模的限制,当某个社区的节点规模小于某个阈值时,停止对它继续进行分裂.

综合以上的分析,本文提出一种二阶段软件聚类方法,该方法主要包括 2 个步骤:①基于软件网络结构模式的聚类.②限制最小社区规模的聚类.基于 GN 算法,在限制社区最小规模的前提下,对软件进行聚类.

2.1　基于软件网络结构模式的聚类

2.1.1　卫星结构

网络中友邻节点个数等于 1 的节点称为卫星节点,而对应的友邻节点称为集散节点.卫星节点要和外界进行交互,必须通过集散节点.因此,可以将卫星节点和它所连接的集散节点合并成一个节点,如图 1 所示.



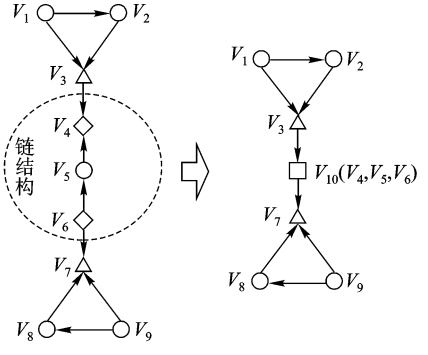
◇—集散节点;□—复合节点;△—卫星节点;○—其它节点.

图 1　卫星结构聚类实例

2.1.2　链结构

本文将网络中友邻节点个数为 2 的节点称为链型节点.如果多个链型节点彼此相连,这样的结构称为链.链两端的节点叫做端点,而它们所连接的非链型节点,称为锚点.

链结构在软件中只负责消息的包装和传递.因此,可以将链结构聚类成 1 个复合节点,复合节点和锚点的连接关系与链端点和锚点的连接关系保持不变,如图 2 所示.



◇—端点;□—复合节点;△—锚点;○—其它节点.

图 2　链结构聚类实例

2.1.3　拓扑相似结构

软件系统中,与外界交互模式相似的实体往往属于同一个模块.因此,可以根据节点之间的协作相似度进行软件聚类.

1) 节点的相似度评价

定义 1 节点 V_i 和 V_j 的前驱节点相似度为

$$\mathrm{Sim}_{\mathrm{pre}}(V_i,V_j)=\begin{cases}\dfrac{|\Gamma_{\mathrm{pre}}(V_i)\cap\Gamma_{\mathrm{pre}}(V_j)|}{|\Gamma_{\mathrm{pre}}(V_i)\cup\Gamma_{\mathrm{pre}}(V_j)|} & |\Gamma_{\mathrm{pre}}(V_i)\cup\Gamma_{\mathrm{pre}}(V_j)|>0\\ 1 & |\Gamma_{\mathrm{pre}}(V_i)\cup\Gamma_{\mathrm{pre}}(V_j)|=0\end{cases}$$

定义 2 节点 V_i 和 V_j 的后继节点相似度为

$$\mathrm{Sim}_{\mathrm{suc}}(V_i,V_j)=\begin{cases}\dfrac{|\Gamma_{\mathrm{suc}}(V_i)\cap\Gamma_{\mathrm{suc}}(V_j)|}{|\Gamma_{\mathrm{suc}}(V_i)\cup\Gamma_{\mathrm{suc}}(V_j)|} & |\Gamma_{\mathrm{suc}}(V_i)\cup\Gamma_{\mathrm{suc}}(V_j)|>0\\ 1 & |\Gamma_{\mathrm{suc}}(V_i)\cup\Gamma_{\mathrm{suc}}(V_j)|=0\end{cases}$$

其中，$\Gamma_{\mathrm{pre}}(V_i)$ 和 $\Gamma_{\mathrm{suc}}(V_i)$ 分别为节点 V_i 的前驱和后继节点集合；$\Gamma_{\mathrm{pre}}(V_j)$ 和 $\Gamma_{\mathrm{suc}}(V_j)$ 分别为节点 V_j 的前驱和后继节点集合；$|A|$ 为集合 A 中的元素个数(下同).

定义 3 节点 V_i 和 V_j 的相似度为

$$\mathrm{Sim}(V_i,V_j)=\mathrm{Sim}_{\mathrm{pre}}(V_i,V_j)\times\mathrm{Sim}_{\mathrm{suc}}(V_i,V_j)$$

2) 基于相似度显著性的节点聚类

定义 4 节点 V_i 的平均相似度为

$$\mathrm{AvgSim}(V_i)=\frac{\sum_{V_j\in\psi(V_i)}\mathrm{Sim}(V_i,V_j)}{|\psi(V_i)|}$$

其中 $\psi(V_i)$ 为与节点 V_i 的相似度大于 0 的节点集合.

定义 5 节点 V_i 的相似度标准差为

$$\mathrm{StdevSim}(V_i)=\sqrt{\frac{\sum_{V_j\in\psi(V_i)}[\mathrm{Sim}(V_i,V_j)-\mathrm{AvgSim}(V_i)]^2}{|\psi(V_i)|-1}}$$

当 $\psi(V_i)=1$ 时，$\mathrm{StdevSim}(V_i)=0$.

定义 6 节点 V_j 相对 V_i 的相似度显著性为

$$\mathrm{ZScore}(V_j,V_i)=\frac{\mathrm{Sim}(V_j,V_i)-\mathrm{AvgSim}(V_i)}{\mathrm{StdevSim}(V_i)}$$

当 $\mathrm{Sim}(V_i,V_j)=1$ 或者 $\mathrm{ZScore}(V_i,V_j)$ 与 $\mathrm{ZScore}(V_j,V_i)$ 同时大于等于某个阈值 ε 时，称 2 个节点显著相似. 显著相似的节点对 (V_i,V_j) 被添加到候选的合并队列 Ω 中.

初始状态下，Ω 中的元素可能存在交集，本文采用如下策略来确定最大可合并的显著节点集合：设 γ_s 和 γ_t 是 Ω 中的 2 个显著相似的节点集合，$|\gamma|$ 表示集合中元素的个数.

① 如果 $\gamma_s\cup\gamma_t$ 中的任意 2 个节点 V_i 和 V_j，满足 $\mathrm{Sim}(V_i,V_j)=1$ 或者 $\mathrm{ZScore}(V_i,V_j)\geqslant\varepsilon$，那么从 Ω 中删除 γ_s 和 γ_t，并将 $\gamma_s\cup\gamma_t$ 添加到 Ω 中.

② 如果 $\gamma_s\cup\gamma_t$ 中，存在 2 个节点 V_i 和 V_j，$\mathrm{Sim}(V_i,V_j)\neq1$ 而且 $\mathrm{ZScore}(V_i,V_j)<\varepsilon$，则按如下准则依次处理：

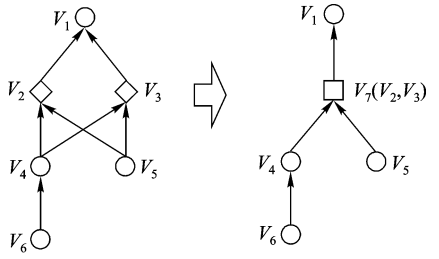
当 $|\gamma_s|>|\gamma_t|$，则从 Ω 中删除 γ_t；

当 $|\gamma_s|=|\gamma_t|$，并且 $\sum_{V_i,V_j\in\gamma_s,i\neq j}\mathrm{Sim}(V_i,V_j)/|\gamma_s|>\sum_{V_i,V_j\in\gamma_t,i\neq j}\mathrm{Sim}(V_i,V_j)/|\gamma_t|$，则从 Ω 中删除 γ_t；

当 $|\gamma_s|=|\gamma_t|$，并且 $\sum_{V_i,V_j\in\gamma_s,i\neq j}\mathrm{Sim}(V_i,V_j)/|\gamma_s|=\sum_{V_i,V_j\in\gamma_t,i\neq j}\mathrm{Sim}(V_i,V_j)/|\gamma_t|$，则从 Ω 中随机删除 γ_s 或 γ_t.

③ 重复步骤①～②，直到集合 Ω 不再变化.

图 3 给出了一个拓扑相似结构的聚类实例. 其中，节点 V_2 和 V_3 的相似度为 1，因此被合并成复合节点 V_7.



◇—显著相似节点; □—复合节点; ○—其它节点.

图 3 拓扑相似结构聚类实例

3) 基于软件结构模式的聚类算法

综合前面的分析，本文提出基于软件结构模式的聚类算法. 算法包括以下步骤：

① 反复检测网络中的卫星结构，将卫星节点和对应的集散节点合并为一个复合节点；

② 反复检测网络中的链结构，将链结构合并为一个复合节点；

③ 反复检测网络中的显著相似的节点集，并合并成一个复合节点；

④ 重复步骤①～③，直到网络不再变化.

2.2 限制最小社区规模的聚类

限制最小社区规模的软件聚类算法，是在 GN 算法的基础上，增加社区规模的限制，禁止 GN 算法分裂出规模小于特定阈值的社区. 该算法包括以下步骤：

1) 将基于软件结构模式聚类后的有向网络转换成无向网络，并作为初始网络 G_{ud}；

2) 计算网络中每条边的边介数，并对所有边按介数进行排序；

3) 选择介数最大的边，并尝试将它从网络中删除；

4) 如果去边后,网络中存在某个社区,它所包含的原始节点个数小于某个阈值,则将删除的边重新添加到网络中,选择介数次大的边进行删除;

5) 如果已经遍历完所有的边,则跳到步骤 8);否则,重复步骤 4),直到成功删除一条边;

6) 计算当前划分下 G_{ud} 的模块度 Q;

7) 将去边后的网络作为 G_{ud},重复步骤 3)~7);

8) 选择模块度 Q 最大的候选划分,并将划分中的复合节点还原成真实网络中的原始节点;

9) 将步骤 8)的结果作为系统的最终划分.

3 实例分析

为了验证二阶段软件聚类方法的有效性,本文将该方法与 Bunch 方法[2]、单纯使用 GN 算法的聚类结果进行对比.

3.1 软件网络的定义

软件中的每个类和接口都映射成网络中的一个节点.类/接口间的下列关系映射成网络中的边:①类/接口 A 继承类/接口 B,那么 $A\rightarrow B$;②如果类 A 实现接口 B,那么 $A\rightarrow B$;③类/接口 A 包含一个成员变量,变量类型是类/接口 B,那么 $A\rightarrow B$;④类/接口 A 中存在一个成员函数,它的某个参数类型是类/接口 B,那么 $A\rightarrow B$;⑤类/接口 A 的成员函数中包含一个类/接口 B 的成员变量,或者调用 B 中的方法,那么建立 $A\rightarrow B$.

以下给出一段Java源代码,对应的依赖网络模型如图 4 所示.

```
public class Painter extends Person implements
Paint-Skill {
    private Address add;
    public void paint (Furniture furn) {
        Logger log;
        // paint furniture and save log info;
    }
}
```

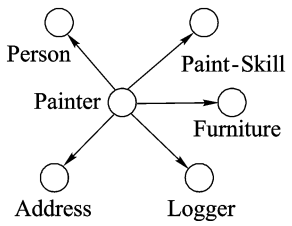


图 4　Java 源代码片段以及对应的依赖网络

3.2 实验设计

本文的实验软件包括:①Catalina.jar:开源 Web 应用服务器 Tomcat4.0.1 的 Servlet 容器[9].②jedit.jar:开源程序编辑器 JEdit4.2 的核心包[10].

所有实验都只针对软件网络中的最大弱连通子图进行.相似度的显著性阈值 ε 设定为 3.0,社区的最小规模为 3.软件的参考划分以软件包的层级结构为基础,每个子包对应软件中的一个模块[11].聚类结果和参考划分之间的相似度评价指标包括:MoJo[12],MoJoPlus[13],EdgeMoJo[14] 和 MoJoFM[15].其中 MoJoFM 是相似度指标,其它 3 个指标是距离指标.距离越小、相似度越高表示聚类的效果越好.

3.3 结果分析

表 1 中列出了 2 个开源软件包中最大连通子图的基本信息.在基于软件结构模式进行聚类后,网络规模平均减少约 30%.

表 1　软件网络的基本信息

软件网络	原始网络			基于结构模式聚类后的网络		
	节点数	边数	模块数	节点数	边数	复合节点数
Catalina	226	1137	17	168	697	31
JEdit	796	3883	23	537	2232	108

表 2 列出了 3 种方法的聚类结果,以及与参考划分之间的相似度得分.每个度量指标包含 2 组数据,分别表示从聚类结果到参考划分的相似度/距离得分,以及从参考划分到聚类结果的相似度/距离得分.本文在使用 Bunch 方法时,选择爬山法进行优化,所有参数使用缺省值.

表 2　3 种方法的聚类结果对比

软件网络	方法	模块数	单节点模块数	MoJo	EdgeMoJo	MoJoFM/%	MoJoPlus
Catalina	Bunch	11	0	136/107	208/135	34.93/50.23	42/42
	GN	99	81	147/136	147/244	29.67/37.04	105/105
	二阶段聚类	19	0	96/102	100/141	54.07/51.66	47/47
JEdit	Bunch	34	0	273/475	366/758	64.77/37.66	84/84
	GN	52	0	274/329	340/567	64.65/57.66	81/81
	二阶段聚类	29	0	304/258	400/406	60.77/66.62	64/64

从实验结果上看,可以得出结论:

1) 使用GN算法进行聚类时,确实有可能产生单节点社区,但这与网络的结构有关;

2) 以GN算法为代表的复杂网络社区划分算法,同样可以应用到软件聚类中,但需要根据软件的领域特点进行相应的调整;

3) 由于在第1阶段聚类中,减少了约30%的网络规模,使得二阶段聚类算法的效率要明显优于直接使用GN,同时可以避免单节点社区的出现;

4) 使用本文提出的二阶段聚类方法,总体上要优于Bunch方法和GN算法.

4 结 论

本文将复杂网络的社区划分算法引入到软件聚类中,针对GN算法计算量大、可能生成小社区的缺陷,结合软件网络的结构特征提出了一种二阶段软件聚类方法:首先根据软件网络的结构模式进行聚类,其次使用限制社区规模的GN算法进行聚类.

通过考察该算法在2个开源软件包的聚类中的表现,证明本方法可以适用于大型软件的聚类分析,聚类效果在总体上要优于Bunch方法和GN算法.

参考文献(References)

[1] Tzerpos V, Holt R C. ACDC: an algorithm for comprehension-driven clustering[C]// 7th Conference on Reverse Engineering (WCRE-2000). Brisbane, Australia: IEEE, 2000: 258-267

[2] Mitchell B S, Mancoridis S. On the automatic modularization of software systems using the Bunch tool[J]. Transactions on Software Engineering, 2006, 32(3): 193-208

[3] Andritsos P, Tzerpos V. Software clustering based on information loss minimization[C]// 10th Working Conference on Reverse Engineering (WCRE-2003). Victoria, BC, Canada: IEEE, 2003: 334-344

[4] Wu J, Hassan A E, Holt R C. Comparison of clustering algorithms in the context of software evolution[C]// 21st IEEE International Conference on Software Maintenance (ICSM 2005). Budapest, Hungary: IEEE, 2005: 525-535

[5] 李兵,马于涛,刘婧,等. 软件系统的复杂网络研究进展[J]. 力学进展, 2008, 38(6): 805-814
Li Bing, Ma Yutao, Liu Jing, et al. Advances in the studies on complex networks of software systems [J]. Advances in Mechanics, 2008, 38(6): 805-814(in Chinese)

[6] Valverde S, Cancho R F, Sole R V. Scale-free networks from optimal design[J]. Europhysics Letters (EPL), 2002, 60(4): 512-517

[7] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the National Academy of Sciences of the United States of America, 2002, 99(12): 7821-7826

[8] Newman M E, Girvan M. Finding and evaluating community structure in networks[J]. Physical Review E, 2004, 69(2): 026113-1-026113-15

[9] Tomcat[CP/OL]. [2008-11-10]. http://tomcat.apache.org/

[10] Jedit[CP/OL]. [2008-11-10]. http://www.jedit.org/

[11] Shtern M, Tzerpos V. A framework for the comparison of nested software decompositions[C]// 11th Working Conference on Reverse Engineering (WCRE-2004). Delft, Netherlands: IEEE, 2004: 284-292

[12] Tzerpos V, Holt R C. MoJo: a distance metric for software clusterings[C]// 6th Working Conference on Reverse Engineering (WCRE-1999). Atlanta, GA, USA: IEEE, 1999: 187-193

[13] El-Ramly M, Iglinski P, Stroulia E, et al. Modeling the system-user dialog using interaction traces[C]// 8th Working Conference on Reverse Engineering (WCRE-2001). Stuttgart: IEEE, 2001: 208-217

[14] Wen Z, Tzerpos V. Evaluating similarity measures for software decompositions[C]// 20th IEEE International Conference on Software Maintenance (ICSM-2004). Chicago, IL, USA: IEEE, 2004: 368-377

[15] Wen Z, Tzerpos V. An effectiveness measure for software clustering algorithms[C]// 12th IEEE International Workshops on Program Comprehension (IWPC-2004). Bari, Italy: IEEE, 2004: 194-203

(编 辑:赵海容)