ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.6, June 2009, pp.1487−1498
doi: 10.3724/SP.J.1001.2009.03417

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

UML类图层次化自动布图算法*

王晓博+, 王 欢, 刘 超

(北京航空航天大学 软件工程研究所,北京 100191)

Automatic Hierarchical Layout Algorithm for UML Class Diagram

WANG Xiao-Bo+, WANG Huan, LIU Chao

(Software Engineering Institute, BeiHang University, Beijing 100191, China)

+ Corresponding author: E-mail: boby@sei.buaa.edu.cn, http://www.sei.buaa.edu.cn

Wang XB, Wang H, Liu C. Automatic hierarchical layout algorithm for UML class diagram. *Journal of Software*, 2009,20(6):1487−1498. http://www.jos.org.cn/1000-9825/3417.htm

Abstract: UML class diagrams are helpful for understanding complicated object-oriented software systems. The reasonable placement of diagram elements can make class diagrams more readable and understandable. As inheritance is regarded as a hierarchical relation, the hierarchical layout method is usually adopted to draw UML class diagrams. Because the domain specific knowledge and drawing criteria related to class diagrams must be considered in the layout of diagrams, general hierarchical layout algorithms for nested digraphs should be extended according to these criteria. But existing hierarchical layout algorithms for class diagrams cannot handle the nested relations among packages, classes, and interfaces, and existing compound layout methods for digraphs cannot be used directly to draw class diagrams. Layout criteria are selected based on the knowledge of graph drawing aesthetics, UML class diagram semantics and software visualization. In addition, the nested constraints in rank assignment, edge crossing, and coordinate assignment of hierarchical layout method were also analyzed in this paper. Then, existing hierarchical layout algorithm was extended to cope with nested graphs according to nested criteria. Experiment results show that the drawings of the reversed class diagrams are more readable and understandable with proper hierarchies, nested relations, less crossings and optimal drawing area.

Key words: layout algorithm; software visualization; UML class diagram; reverse engineering

摘 要: UML 类图能够有效地帮助软件工程师理解大规模的软件系统,而优化图元的空间布局可以增强类图的可读性和可理解性.由于类图中继承关系具有明显的层次特性,因此类图自动布局大多采用层次化的布图算法.此外,类图布局需要考虑相关的领域知识以及绘制准则,因而通用嵌套有向图层次化布局算法不能直接用于类图的绘制,它们必须加以扩展.但是,已有的类图层次化方法并没有考虑类图中图元的嵌套关系,这将导致自动布局方法不能处理类图中包与类、接口之间的包含关系.在考虑图绘制美学、UML 类图绘制以及软件可视化等相关知识的基础上,选取了一组布图准则并分析了嵌套关系在层次分配、层内排序和坐标分配中引入的约束,通过在层次化方法的主要步骤中引入嵌套约束,提出了嵌套有向图层次化布图算法.实验结果表明,扩展的布图算法能够适应于逆向类

* Supported by the National Natural Science Foundation of China under Grant No.90718018 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z176 (国家高技术研究发展计划(863))

Received 2008-04-03; Accepted 2008-07-02

图的绘制,它具有层次清晰、支持图元嵌套、交叉数目少以及占用面积小等优点.

关键词: 布图算法;软件可视化;UML 类图;逆向工程

中图法分类号: TP311 **文献标识码:** A

UML类图是一种描述软件静态结构的有效方法,它在软件正向建模、逆向工程和程序理解等领域都发挥着重要的作用.Tilley[1]通过实验指出,UML类图可以帮助软件工程师理解大规模的软件系统,但其有效性主要取决于UML的语法和语义、图元的空间布局和领域知识等方面,其中空间布局在增强类图的可读性以及可理解性方面显得尤为关键.现有的可视化建模工具(如IBM RSA,rational rose,borland together,eclipse GMF等)都提供了类图自动布局以及从程序中逆向绘制类图的功能,而目前UML类图自动布局所采用的布图算法主要分为拓扑-形状-度量[2,3]、层次化[4−6]、力学制导[7]和遗传算法[8]等方法.由于类图中的继承关系具有明显的层次特征,为了使布局结果的层次更为清晰,在设计类图布图算法时,通常将聚集关系和实现关系也作为具有层次特征的边来处理[4,9,10],所以,基于层次化的类图布局方法应用较广.然而,目前的类图层次化方法没有考虑类图中包(package)之间的嵌套关系,从而导致布图算法不能处理类图中包与类、接口之间的包含关系.已有的支持嵌套图布局算法[11,12]并没有考虑到嵌套类图的特点,而且它们都根据不同领域的需求进行了扩展和改造,因此不能直接用于类图的自动布局.本文将主要讨论如何扩展层次化布图算法以支持含有嵌套图元的逆向类图绘制.

类图的可读性与可理解性主要通过布局约束来限定,而这些约束来自于通用的图绘制(graph drawing)美学、人机交互、软件可视化以及软件工程等方面.约束之间很容易产生冲突,如图的高度最小化与图纵横比调整、边的长度优化与边交叉最小化等.因此,如何选择合适的布图约束对于布图算法的设计非常重要.Sun[13]对类图布局约束进行了系统的分类和整理,本文依据逆向绘制类图的特点从中选取了一组布局约束,并在Gansner[14]层次化算法的基础上提出了支持图元包含的嵌套有向图层次化算法.层次化布图主要分为层次分配、层内排序和坐标分配 3 个步骤,如何将图元之间的嵌套关系转化为布图约束并应用到层次化布图的关键步骤中,是本文要解决的核心问题.

本文第 1 节描述逆向绘制类图的布局约束.第 2 节分析图元嵌套关系对层次化类图布局所产生的影响.第 3 节介绍嵌套有向图布图算法.第 4 节是实验结果分析及讨论.第 5 节给出结论.

1 逆向绘制类图的布局约束

在逆向绘制的类图中,节点类型分为包、类和接口,边的类型分为继承、聚集、实现和关联.Sun[13]整理的约束分为 9 类,共 14 条,本文根据类图可读性与可理解性从中选取最主要的约束并将其归纳为以下 5 条规则:

A1. 具有明显的层次化结构并能展示图元之间的嵌套关系,所有层次边(具有层次特性的边,如继承、聚集、有向关联)的方向尽可能地一致.文中默认层次方向为自下而上,即类图为上行图(upward graph).

A2. 减少边交叉的数目和边中拐点的数目,提高边的连贯性.

A3. 最小化边的长度,尽量缩短相连节点之间的距离.

A4. 避免边与包图元之间的交叉以及包图元之间的重叠,这将会导致一条边与多个包内的子图元交叉.

A5. 减小结果图所占用的面积,具有合理的纵横比.

对于满足规则A2~A5,目前已有行之有效的布图算法[7,14],但是规则A1 的引入对规则A2~A5 都造成了直接影响,因为传统的基于Sugiyama[15]层次化算法都不能处理图元嵌套的情况.Sugiyama[12]给出了一种绘制普通复合图的方法,但是它所处理的普通复合图与类图不同,算法中并不区分节点和边的类型,而且在层次化和边交叉最小化时不能处理UML类图特有的节点顺序有效性.为了深入讨论具有嵌套图元特性的布局约束,本文首先定义了支持嵌套图元的类图以及布图算法中用到的基本概念,进而把规则A1~A5 转化为数学语言描述的约束.

1.1 类图定义

定义 1(有向图). $G=\langle V,E,\phi\rangle$为有序三元组,其中,$V$为节点集(非空),$E$为边集,而$\phi$是$E$到$V$中元素有序对簇$V\times V$

的函数,称为关联函数.V中元素称为节点(vertex),E中元素称为边(edge),ϕ刻画了边与节点之间的关联关系.令$v\in V$为G中的节点,记$d(v)$为v的度数,$d^+(v)$为v的出度,$d^-(v)$为v的入度.

类图中除了具有方向性的继承边、聚集边和实现边之外,还存在无方向性的关联边,为了应用有向图层次布图算法,本文通过比较关联边两个端点的度数指定了它的方向($e=(u,v)$,如 $d(u)\geq d(v)$,指定 $u\to v$ 为 e 的方向,否则指定 $v\to u$ 为其方向).下面通过在有向图的基础上添加节点包含关系树来定义嵌套有向图.

定义 2(嵌套有向图). $G'=\langle V,E,\phi,T\rangle$,节点包含关系树$T=\langle V^T,E^T\rangle$,其中,$V=V^T$,$E^T\subseteq V^T\times V^T$.$E^T$中的每一条边$e=(u,v)$都在嵌套有向图$G'$中定义了一个包含关系,它表示$u$包含于$v$,$v$称为$u$的父节点,而$u$称为$v$的孩子节点.

定义 3(类图). 类图是一个嵌套有向图 $C=\langle V,E,\phi,T\rangle$,其中,$V$ 中节点的类型有包、类和接口,E 中边的类型有继承关系、聚集关系、实现关系和关联关系,图中节点根据嵌套约束分为两类,即包含节点(包节点)和普通节点(类节点和接口节点).包含关系树 T 中的边 $e=(u,v)$表示包节点之间或者包节点与普通节点之间的包含关系,这里,u 只能是包节点,而 v 可以是包节点或者普通节点.

1.2 类图布局约束定义

为了描述布局约束,首先要对布图算法的中间过程加以定义.层次化布图算法主要包括层次分配、层内排序和坐标分配 3 个步骤.下面分别给出每个步骤的具体定义.

定义 4(层次分配). $G=\langle V,E,\phi,T,R\rangle$为分层有向图,其中,$R$为$V$到有限自然数集$\{1,\ldots,k\}$的映射,即把集合$V$分成$k$个节点集合$V_1,\ldots,V_k$.定义$R(v)$为节点$v$的层号,$length(e=(u,v)\in E)=R(v)-R(u)$表示边的长度.为了保证边的长度的有效性,规定$length(e)\geq\delta$,$\delta$为边的最短长度,本文默认为 1.这里引入的层号将在坐标分配时决定节点的纵坐标.

定义 5(层内排序). $G=\langle V,E,\phi,T,R,O\rangle$为排序后的分层有向图,$V_i(1\leq i\leq k)$为$G$中第$i$层节点的集合,其中,$O$为$V_i$中节点$v$到有限自然数集$\{1,\ldots,|V_i|\}$的映射,$O(v,i)$为节点$v$在第$i$层内的层内序号,它将在坐标分配中决定节点的横坐标.

定义 6(坐标分配). $G=\langle V,E,\phi,T,L\rangle$为分配坐标后的类图,其中,$L$ 为 V 到自然数对集 $N\times N$ 的映射,即为每一个节点分配横、纵坐标.令 v 是图中的节点,记 $x(v)$为 v 的横坐标,$y(v)$为 v 的纵坐标,$width(v)$为 v 的宽度,$height(v)$为 v 的高度.类图中普通节点的宽度和高度是根据其内容预先计算的,而包节点需要在确定其子节点的位置布局后才能计算其尺寸.

接下来,本文将根据以上的基本定义给出布局约束的数学描述.规则A1 要保证所有的包节点必须位于其所包含的节点上方,且所有被包含的节点不能超出该包节点的左右边界和下边界.文中设定坐标原点为左上角,横轴方向为从左到右,而纵轴方向为从上到下.令p为一个包节点,$C=child^*(p)\subseteq V^T$为包含关系树$T$中$p$的所有子孙节点,则对于集合$C$中每一个节点$c$都有:

$$R(p)<R(c),left(p)<left(c),right(p)>right(c),bottom(p)>bottom(c).$$

同时,规则 A1 还要求结果图为上行图,即尽量使具有层次性的边 $e=(u,v)$都满足约束 $R(v)>R(u)$.

层次化布图算法大多采用逐层最小化方法将层次图转化为一组两层图(两层图,即只由相邻层节点组成).为了应用两层有向图边交叉最小化算法,图中所有边的长度必须为 1,常用的方法是在跨越多层的边中引入拐点来分解它,即对于$e=(u,v)$,$R(v)-R(u)>1$ 这样的边中引入$R(v)-R(u)-1$ 个拐点.规则 A2 中,边交叉最小化是通过层内排序来达到的.边交叉是由相邻层之间的边起点和终点的相对顺序不同所引起的[7].如图 1 所示,边(u,x)与边(v,y)相交,当且仅当$(O(u,1)-O(v,1))(O(x,2)-O(y,2))<0,u\neq v,x\neq y$,因此,规则 A2 要求布图算法尽可能地减少这种交叉.根据定义 4,$length(e)=R(v)-R(u)$,则拐点的数目等价于 $length(e)-1$,因此,规则 A2 中拐点数目最小化与规则 A3 中边的长度最小化可以统一表述为 $\mathrm{Min}\left(\sum_{e\in E}length(e)\right)$.规则 A4 可以通过添加边界边转化为边交叉最小化的问题,具体将在第 3.2 节中加以介绍.而规则 A5 对绘图结果所占用面积的约束可以表述为图的宽度与其高度乘积的最小化.

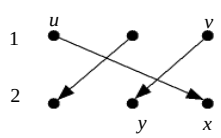


Fig.1 Crossing example
图 1 边交叉示例

令 $Width=\underset{v\in V}{\mathrm{Max}}(x(v))-\underset{v\in V}{\mathrm{Min}}(x(v))$ 为图的宽度, $Height=\underset{v\in V}{\mathrm{Max}}(y(v))-\underset{v\in V}{\mathrm{Min}}(y(v))$ 为图的高度,则结果所占的最小面积为 $\mathrm{Min}(Width\times Height)$.

2 图元嵌套引入的问题

在给出类图和层次化布图主要步骤的具体定义后,下面将进一步讨论图元嵌套给传统层次化类图布局带来的问题,并给出相应的解决方法.根据定义 3 的描述,类图是一种嵌套有向图,因此我们可以对它应用层次化有向图布图算法.本文主要在Gansner[14]有向图层次化布图算法的基础上讨论如何支持嵌套有向图的布局.图元之间的嵌套关系直接影响了节点的层次分配,包节点必须位于它所包含节点的上方.此外,由于没有包含关系的包节点之间不能相互重叠,边交叉最小化时经常用的交换节点顺序操作就可能导致包节点相互重叠.第 2.1 节和第 3.1 节将从层次分配和节点排序两个方面分别讨论这些问题.

2.1 确定嵌套图元的层次

层次分配的目的是在遵循布图约束的前提下把节点划分到不同的层中去,除了边长度最小化的约束外,层次分配时还要考虑类图中的嵌套约束.通过观察任意两个节点在垂直方向的相对位置与它们的层号之间的关系,可得定理 1.

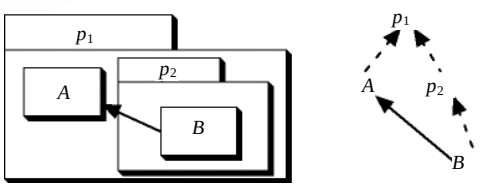
定理 1. 如 $R(u)>R(v)$,则 u 位于 v 的上方;反之,u 位于 v 的下方.u 和 v 的垂直位置相同当且仅当 $R(u)=R(v)$.

根据规则 A1,所有的包节点必须位于其子孙节点的上方,同时要尽可能地使具有层次特性的边的方向向上.由于在层次化布局方法中层次性边起主导作用,因此在层次分配时只需考虑具有层次特性的边即可.定义 4 给出了边长度的定义,并且规定 $length(e\in E)\geq\delta=1$,由此限定可得定理 2.

定理 2. 在分层有向图中,如果有层次边 $e=(u,v)$,那么 v 在 u 的上方.

证明:由边长定义可知,$length(e=(u,v))=R(v)-R(u)$且 $length(e)\geq 1$,因此 $R(v)-R(u)\geq 1\Rightarrow R(v)>R(u)$.根据定理 1 可得,$v$ 位于 u 的上方. □

根据定理 2,如果在包节点及其子节点之间添加一条表示包含关系的层次辅助边,就可以保证包节点位于其子节点的上方.具体方法如图 2 所示,把类图$C=\langle V,E,\phi,T\rangle$包含关系树中的每一条边都作为层次辅助边添加到层次分配图中去,图 2(b)为图 2(a)所对应的嵌套有向图,其中虚线边表示包含关系的层次辅助边.图 2 所示的方法能够处理所有节点之间的嵌套关系.如果两个包节点之间没有嵌套关系,而它们的子孙节点却存在层次关系(如图 3(a)、图 3(b)为图 3(a)对应的嵌套有向图),那么分层时为了减小整图的高度,会把它们分配到相邻层中(如图 3(c)所示).当包含关系树上某一层中包节点的数目过多时,图 3(c)的分配策略就会导致结果图的纵横比失调,而且包节点之间的依赖关系也不能通过层次化的方法表现出来(如图 3 中,p_2对p_1的依赖关系,p_2的最佳摆放位置应位于p_1及其所有子孙节点的下方).类似于前面确定包节点位置的方法,可以通过在两个相互依赖的包节点之间添加层次辅助边来修正它们的垂直位置.以图 3 为例,p_2和p_1的辅助边只能保证p_2位于p_1的下方(如图 3(c)所示);为了能够达到图 3(a)的效果,可以在p_1中添加一个底部节点p_1b(如图 3(d)所示),并在p_1b和p_1及其所有子节点之间添加层次辅助边,同时在p_1b与p_2之间也添加包依赖辅助边.



(a) (b)

Fig.2 Add inclusion edge

图 2 添加包含辅助边

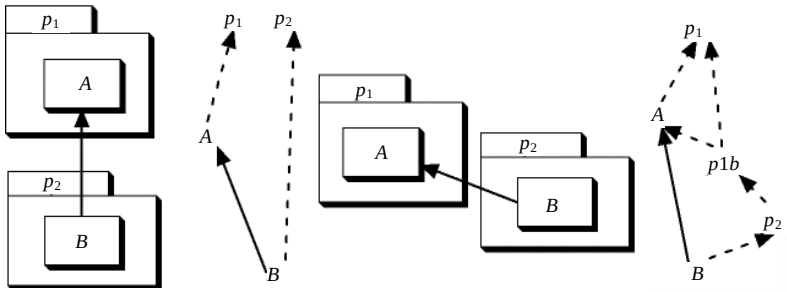


Fig.3 Add package dependence edges

图 3 添加包依赖辅助边

2.2 保证节点顺序有效性

为了在分层嵌套有向图中应用逐层边交叉最小化算法,所有跨越多层的边(即$length(e)>1$)都将通过插入辅助节点转化为长度为 1 的边.同时还要删除在层次分配中添加的包含辅助边和包依赖辅助边,因为它们只在层次分配中用于限定节点的垂直位置.层次有向图中的边交叉只存在于相邻层之间(如图 1 所示),而且整个图中只存在边与边之间的交叉;但在嵌套层次有向图中,由于节点之间存在包含关系,图中的边可能与包含节点的两个边界同时相交(如图 4 中边EF与p_2的左、右边界同时交叉).边交叉最小化算法的主要思想是交换存在交叉的节点顺序,然而,嵌套关系却对这种交换添加了新的约束,即两个没有包含关系的包含节点在任意一层的相对位置都是相同的.如图 4 中边BC与AD之间的交叉,如果第 1 层中节点的顺序确定,那么p_1位于p_2的左边,但是它不能通过交换第 2 层中A与C的顺序来消除,因为交换A,C则意味着在第 2 层中p_1位于p_2的右边.为了尽量避免边与包含节点之间的交叉以及保证包含节点在不同层次间相对顺序的一致性,下面给出层内有效、层间有效以及边与包含节点相交的详细定义.

定义 7(层内有效). 令$G=\langle V,E,\phi,T,R,O\rangle$为排序后的分层嵌套有向图,$c$是图中的包含节点,$V_i$为第$i$层的节点集合,$V_c=\{v\mid v\in child^*(c)\}$为$c$的所有子孙节点,而$V_{(c,i)}=\{v\mid v\in(V_i\cap V_c)\}$为$c$在第$i$层中的所有子孙节点,则$\mathcal{R}(c,i)=[O_{\mathrm{Min}}(V_{(c,i)},i),O_{\mathrm{Max}}(V_{(c,i)},i)]$为$V_{(c,i)}$层内序号的范围.图$G$是层内有效的当且仅当对于图$G$中每一个包含节点$c$都有$\forall v_{v\notin V_c}(O(v,R(v))\notin\mathcal{R}(c,R(v)))$.层内有效保证了一层中每个包含节点的排他性.比如,在图 4 中,如果F位于 $p2$ 的内部,则违反了层内有效的规则.

定义 8(层间有效). 令$G=\langle V,E,\phi,T,R,O\rangle$满足层内有效,$c$为包含节点$\Omega(c)=\{i\mid R_{\mathrm{Min}}(c)\le i\le R_{\mathrm{Max}}(c)\}$为$c$的子孙节点分配的层号范围.图$G$是层间有效的,当且仅当$G$中任意两个包含节点$c_1$和$c_2$,如果它们之间不存在嵌套关系(即$c_1\notin child^*(c_2),c_2\notin child^*(c_1)$),则有$\forall i_{i\in(\Omega(c_1)\cap\Omega(c_2))}(O_{\mathrm{Max}}(c_1,i)<O_{\mathrm{Min}}(c_2,i))$(即$c_1$位于$c_2$的左边)或者$\forall i_{i\in(\Omega(c_1)\cap\Omega(c_2))}(O_{\mathrm{Max}}(c_2,i)<O_{\mathrm{Min}}(c_1,i))$(即$c_1$位于$c_2$的右边).层间有效性保证了任意两个不存在嵌套关系的包含节点的相对位置在任意一层都是一致的.

定义 9(边与包节点的交叉). 令$G=\langle V,E,\phi,T,R,O\rangle$满足层间有效,边$e=(u,v)$与包节点$c$交叉当且仅当$R(v)\ge R_{\mathrm{Min}}(c),R(u)\le R_{\mathrm{Max}}(c),O(u,R(u))>O_{\mathrm{Max}}(c,R(u)),O(v,R(v))<O_{\mathrm{Min}}(c,R(v))$(这里假定$u$在$c$的右边,$v$在$c$的左边,反之,定义$u$在$c$的左边,$v$在$c$的右边).

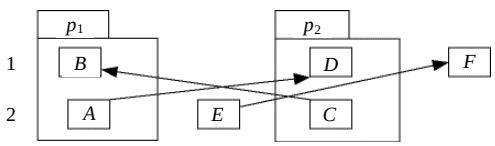


Fig.4 Crossing in nested DAG

图 4 嵌套有向图中的交叉

不同于传统的两层最小化问题,嵌套有向图两层最小化在求解中添加了层内有效和层间有效的约束.具体表现为,当同层两个节点之间发生交换时,需要在交换时判断是否违反这两个有效性.如果两个节点的父节点相同,那么交换它们后仍然满足层内有效性.因此,可以根据这一特点通过同层节点在包含关系树中的相对位置来确定需要被排序的层.Foster[11]在绘制生化路径(biochemical pathway)时提出了聚类图(cluster graph)两层交叉最小化算法框架.本文中的分层嵌套有向图也可以看作是一个聚类图,即所有包含节点的子孙节点组成一簇.根据Foster[11]的方法,下面将给出两层嵌套有向图的定义,其中,边与包节点之间的交叉对应于聚类图中边与簇的交叉.

定义 10(两层嵌套有向图). 令 $G=\langle V,E,\phi,T,R\rangle$为分层嵌套有向图, $GC=\langle V_i\cup V_{i+1},EC,\phi_c,TC\rangle$ 为两层嵌套有向图,其中,V_i与V_{i+1}分别为第i层与第i+1 的节点集合,EC为V_i与V_{i+1}中节点之间的边集合,TC为V_{i+1}中节点及其在T中所有祖先节点组成的包含关系树.

3 嵌套有向图布图算法

通过第 2 节中对图元嵌套关系所引入问题的深入分析,本文分别在第 2.1 节和第 2.2 节中给出了层次分配和层内排序的解决方法,下面将给出具体的算法.为了方便叙述,算法 1 给出了Gansner[14]布图算法的主要步骤.

算法 1. 有向图布图算法.

(1) procedure *draw_directed_graph*();
(2) *rank*(); //为图中的节点分配层号
(3) *ordering*(); //层内节点排序,减小边交叉数目
(4) *positioning*(); //根据节点的层号和层内序号分配纵坐标与横坐标
(5) *make_splines*(); //用曲线绘制含有拐点的边
(6) end.

由于类图中边通常表示为直线或者折线,因此文中不考虑第(5)行绘制曲线边的这一步.接下来,本文按照有向图层次化布局的算法框架分 3 个步骤讨论嵌套有向图布图算法.

3.1 层次分配

层次分配首先根据第 2.1 节中的方法将图元嵌套转换为表示包含关系的层次辅助边,然后在定理 2 的基础上修正具有依赖关系的包图元的相对位置,算法 2 给出了添加包依赖关系的详细步骤.

算法 2. 添加包依赖关系.

输入:类图 $C=\langle V,E,\phi,T\rangle$;

输出:$C=\langle V,E\cup E',\phi,T\rangle$,其中,$E'$为包节点与被依赖包的底部节点之间的依赖辅助边.

(1) procedure *add_package_dependency*();
(2) for each $(e=(u,v)\in E$ & *e.isHierarchical* = true) //遍历 E 中具有层次特性的边
(3) $c=$ *getCommonParent*(u,v,T) //得到 u 和 v 在 T 中最近的公共包节点
(4) $uc=$ *findNearestContainer*(u,c,T) //在 T 中寻找距离 c 最近且包含 u 的节点
(5) $vc=$ *findNearestContainer*(v,c,T) //在 T 中寻找距离 c 最近且包含 v 的节点
(6) *DG.addEdge*(vc,uc) //$DG=\langle VD,ED,\phi_d\rangle$为临时构造的依赖图,把辅助边添加到$DG$中去
(7) end for
(8) *acyclic_and_simplify*(*DG*) //消除 DG 中的对称边、多重边以及循环依赖
(9) $leafnodes=\{v\mid out(v)=0,v\in VD\}$ //从 DG 中出度为 0 的节点出发,把依赖关系插入到 C 中
(10) while ($|leafnodes|\neq 0$);
(11) for each ($s\in leafnodes$);
(12) *C.addEdgeSet*(*inEdges*(s)) //把终点为 s 的依赖边添加到 G 中去
(13) end for

(14) $VD/\{s\}$;*update*(*leafnodes*) //从 DG 中删除节点 *s* 并更新集合 *leafnodes*

(15) end while

(16) end.

算法 2 中,*acyclic_and_simplify*(*DG*)的目的是把包依赖图 *DG* 转化为有向无环简单图.如果节点 *u* 和 *v* 之间存在多条依赖边($UV=\{e=(u,v)\in ED\}$,$VU=\{e=(v,u)\in ED\}$),那么 *u* 和 *v* 的依赖方向将由|*UV*|和|*VU*|决定.如果|*UV*|≥|*VU*|,则确定 $u\to v$ 为其方向,否则确定 $v\to u$ 为依赖方向.

由于层号决定了节点的纵坐标,因此,边的长度会受到层次分配的影响.规则 A2 和 A3 要求图中边的长度总和最小,即 $\mathrm{Min}\left(\sum_{e\in E} length(e)\right)$.因为图中的边具有不同的类型,而且具有层次特性的继承边、聚集边和实现边构成了类图布局的整体层次框架,所以算法需要优先缩短这些边.本文参照Eichelberger[9]的方法为图中的边设定了不同权值,包含边为 1,依赖边为 2,聚集边为 4,继承和实现边为 8.引入边权值后,层次分配中的约束就转化为一个线性规划问题 $\mathrm{Min}\left(\sum_{e\in E}(length(e)\times weight(e))\right)$,$length(e)\ge\delta=1$.文中采用网络单纯形(network simplex)的方法进行求解,实验结果表明,这种方法迭代次数少而且收敛速度较快[9,14].由于类图中可能存在自环、多重边以及环路,而层次分配要求输入的是有向无环简单图,因此首先要对输入的类图进行去环、合并多重边等预处理步骤,本文在实现中采用深度优先遍历的方法去除图中的环路[9].

3.2 层内排序

算法 3 给出了两层嵌套有向图交叉最小化的步骤.由于交换父节点相同的节点不会违反层内有效性,因此,从*TC*(包含关系树)的根节点出发,每次只处理当前节点的直接孩子节点之间的交叉,当*TC*中所有的包含节点都被处理后,算法终止.由于*TC*中每个节点都可能为父节点,在每个父节点处都会产生新的递归遍历,因此算法的时间复杂度为$O(V^2)$.算法 3 通过逐层处理包含关系树将两层嵌套有向图转化为一组两层有向图,而两层有向图最小化常用的求解方法有邻位交换法[7]、Median[16]方法以及BaryCenter[17]方法等.*ApplyCrossingResult*(GC_x)选用了Median方法,因为它在最坏情况下的边交叉数目也不会超过最优结果的 3 倍[7].为了保证每一个GC_x最小化结果与*GC*的映射之间不会相互影响,本文通过后续遍历包含关系树*TC*来保证所有的包含节点按照嵌套层次的深度自底向上被处理.

算法 3. 两层嵌套有向图交叉最小化算法.

输入:两层嵌套有向图 $GC=\langle V_i\cup V_{i+1},EC,\phi_c,TC\rangle$;

输出:对于 *TC* 中所有的包含节点都生成一个两层有向图 $GC_{x\in TC}=\langle V_{i,x}\cup V_{i+1,x},EC_x,\phi_c\rangle$,其中,*x* 为包含节点.

(1) procedure *two_level_cross_reduction*();

(2) for each $(x\in TC\ \&\ types(x)=CLUSTER)$;

(3) $GC_x=\langle VC_x,EC_x,\phi_x,weight\rangle$; //构造两层有向图

(4) $VC_x=V_{i,x}\cup child(x)$; //$VC_x$由第*i*层节点和第*i*+1 层中*x*的直接孩子节点组成

(5) $EC_x=\{(u,v)\in V_{i,x}\times child(x)\mid weight(u,v)>0\}$; // $weight(u,y)=|\{v\in V_{i+1}\cap child^*(x)\mid(u,v)\in EC\}|$

(6) *TwoLevelCrossingReduction*(GC_x); //应用两层最小化算法

(7) *ApplyCrossingResult*(GC_x); //将GC_x最小化的结果映射到原两层有向图*GC*上

(8) end for

(9) end.

层间有效性要求所有的包含节点在每一层的相对顺序都是相同的,而算法 3 并不能保证层间的有效性.本文通过在每一个生成的两层有向图中添加高权值边界边来保证层间有效性.图 5(a)中的节点分为 4 层,第 3 层、第 4 层需要进行边交叉最小化,图 5(b)是包p_1对应的两层有向图,其中,p_2,p_3的子孙节点与第 3 层节点之间的边都将转化为p_2,p_3与第 3 层节点之间的边.由于p_3在第 3 层、第 4 层中都有子孙节点,为了保证层间的有效性,需要对p_3添加边界边(如图 5(c)中与p_3相连的虚线边).在确保层间有效性时,设定边界边的权值为$|EC_x|^2$,即两层有

向图边数目的平方.因为交换之前已经确定相对位置的包含节点会导致边交叉数目超过交叉的上限(边总数的平方),所以高权值边界边能够保证层间有效性.添加边界边不但可以保证层间有效性,而且还能用于减少包含节点与边之间的交叉,区别在于这时边界边的权值设定为 1/2,因为这种交叉不是必须减少的,而且当一条边与左右两条边界边都交叉时,才记为一个交叉.

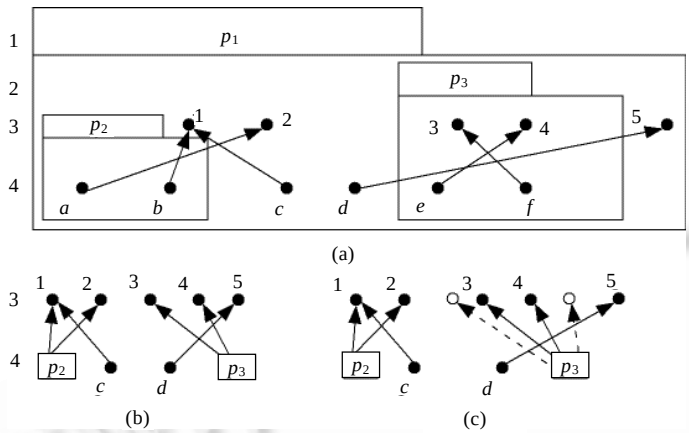


Fig.5 Crossing reduction in nested DAG

图 5 嵌套有向图中的边交叉最小化

3.3 坐标分配

节点的层号和层内序号确定后,布局结果的基本框架就已经确定了.一般有向图可以采用在垂直方向或者水平方向按序号递增的方式分别分配纵坐标和横坐标,但是根据第 1.2 节中的分析,所有包含节点的子孙节点都不能超过它们的边界,同时还要缩短边的长度($e=(u,v), DISTANCE(e)=\sqrt{(x(v)-x(u))^2+(y(v)-y(u))^2}$).由于纵坐标由层号唯一确定,而且分层时已经根据规则 A2 和 A3 进行了优化,因此,分配坐标只需要优化每一条边两个端点的横坐标间距即可.与层次分配类似,规则 A3 中边长度最短的要求在分配横坐标时定义为 $\mathrm{Min}\left(\sum_{e=(u,v)\in E} weight(e)\times|x(v)-x(u)|\right), |x(v)-x(u)|\geq 1$.A5 要求在限定同层节点之间的最小水平间距(NODE_SEP)下尽可能缩小所有节点之间的水平距离,本文在Gansner[14]的基础上同时考虑A1,A3 和A5,给出了嵌套有向图横坐标优化分配算法.

算法 4. 嵌套有向图横坐标优化分配算法.

输入:层内排序后的嵌套有向图 $G=\langle V,E,\phi,T,R,O\rangle$;

输出:分配横坐标后的嵌套有向图 $GL=\langle LV,LE,\phi,TL,L\rangle$.

(1) procedure *x_coordinate_assignment*();

(2) 构造坐标分配图 *GL*,*LV*=*V*,*TL*=*T*;

(3) 为 *GL* 中所有的包添加左、右边界节点,并添加边界水平约束;

(4) for each $(V_i, i\in[0,\mathrm{Max}Rank])$; //遍历所有的层

(5) for $(j=0; j<|V_i|-1; j++)$;

(6) $GL.addEdge(V_i[j], V_i[j+1])$; //添加水平限定关系

(7) end for

(8) end for

(9) for each $(e=(u,v)\in E)$; //遍历 *G* 中所有的边

(10) $c=getCommonParent(u,v,TL)$; //得到 *u* 和 *v* 在 *TL* 中最近的公共包节点

(11) *GL.addEdge*(*u*,*leftBorder*(*c*)) ; //将层次性的边转化为水平方向的约束
(12) *GL.addEdge*(*leftBorder*(*c*),*v*) ;
(13) end for
(14) *SimplexNetworks*(*GL*) ; //对 *GL* 应用网络单纯形算法,分配水平层号
(15) *xCoordinateAssignment*(*GL*) ; //根据水平层号分配横坐标
(16) end.

算法 4 通过添加辅助边把水平距离最小化转化为节点之间水平方向的约束,进而把整个水平坐标优化分配转化为水平层次分配问题.实例如图 6(a)所示,转换后的横坐标分配图如图 6(b)所示,其中,虚线边分别连接p_1,p_2的左右边界,实线边表示坐标分配时加入的水平位置约束边,每一个包含节点的左边界都与其父节点的左边界相连,右边界与其普通子节点相连,对于其类型为包含节点的子节点(如p_1中包含p_2),右边界与子节点的右边界相连.所有的嵌套关系都作为边界节点与包含节点的子节点之间的水平限定边加入到横坐标分配图中,因此根据定理 2,算法 4 显然能够满足嵌套有向图的约束.

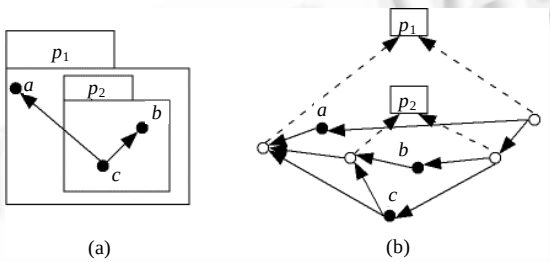


Fig.6 Horizontal coordinates assignment
图 6 横坐标分配

4 实验结果及讨论

本文通过分析一组采用Java语言编写的软件项目并绘制其逆向类图来检验嵌套类图层次化布图算法的有效性.贝尔实验室开发Graphviz[18]采用通用复合图布局算法,可以支持图元之间的嵌套关系,它对复合图的处理类似于Sugiyama[12]的层次化方法.本文将利用Graphviz中的层次化布图工具dot来比较文中类图自动布图算法与通用的嵌套图布图算法.dot并不直接支持类图的自动布局,这里,我们通过生成dot能够识别的复合图脚本文件来驱动它进行布图.因为类图中的边一般为线段或者折线,所以还需要把dot所生成的Beizer曲线边转化为折线边.图 7 是嵌套类图层次化布图算法对Junit 3.8 逆向类图的绘制结果,从图中可以看出,整图的框架层次分明、边交叉较少并且清晰地展现了图元之间的嵌套关系,具有较好的可读性和可理解性.图 8 为dot绘制Junit3.8 的类图,它包含了与图 7 中相同的 3 个包.由于dot不能指定包图元之间的水平最小间隔,因此图 8 中 3 个包的边界都是彼此相连的.图 8 中由于包图元之间没有计算隐含的依赖关系,因此包之间的水平位置只能依靠横向排列来解决,这样就导致整图面积的利用率不高,而且留下了不必要的空白.

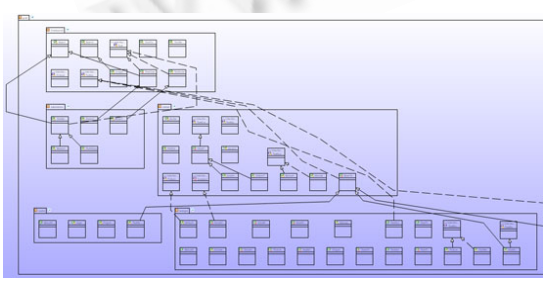


Fig.7 Drawing result of Junit 3.8
图 7 Junit 3.8 的绘制结果

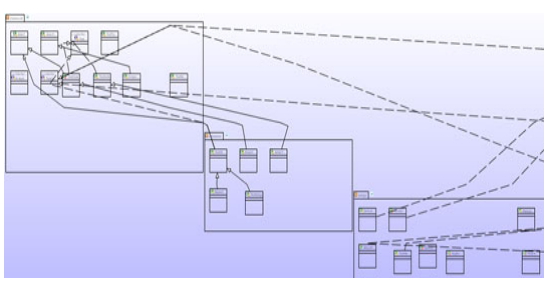


Fig.8 Drawing result of Junit 3.8 by dot
图 8 Dot 绘制的 Junit 3.8 的类图

两幅结果图所取的大小相同,而图 7 可以在占用同样面积的条件下充分利用空间.我们的方法不仅能够根据包之间的依赖关系组织包的纵向布局,而且还能根据图元之间的嵌套关系有效地减少边之间以及边与包之间的交叉数目.Sugiyama[12]在处理包含图元布局时也采用了包含关系树来分配每个节点的层次,而我们在层次分配时不仅考虑了包含关系,同时还把继承、实现等类型的边也都作为层次分配的依据.为了保证嵌套图元之间纵向位置的布局,本文在层次分配中采用深度优先算法去环时,添加了不能反转包含辅助边的约束.

我们分析了 22 个开源软件的类图布局效果,实验结果表明:扩展的嵌套有向图层次分配能够清晰地展示节点之间的层次性边及其嵌套关系,但是由于层次分配时需要去环,因此少量的层次边不能满足上行图约束;扩展的Median两层嵌套有向图交叉最小化算法能够在同时满足层间有效性和层内有效性的约束下达到较好的结果(如图 9 所示,*EE*表示边与边之间的交叉,*EP*表示边与包之间的交叉),而且扩展的交叉最小化方法明显优于邻位交换以及BaryCenter等方法(如图 10 所示).然而由于逐层最小化框架的约束,这种局部优化方法还不能达到最优;基于转换水平约束以及网络单纯形方法的横坐标分配算法能够在很大程度上减少结果图所占用的总面积,但它在时间效率上比Eichelberger[10]扫描线分配方法略逊(平均时间比扫描线算法慢 30%,因为坐标分配时采用了网络单纯形方法,而扫描线算法是基于层次嵌套关系的线性算法).

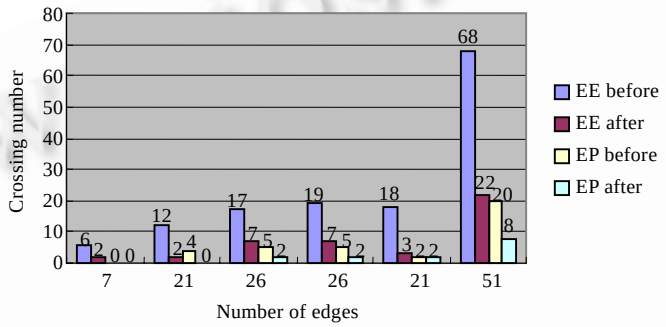


Fig.9 Result of intra-rank sorting

图 9 层内排序结果

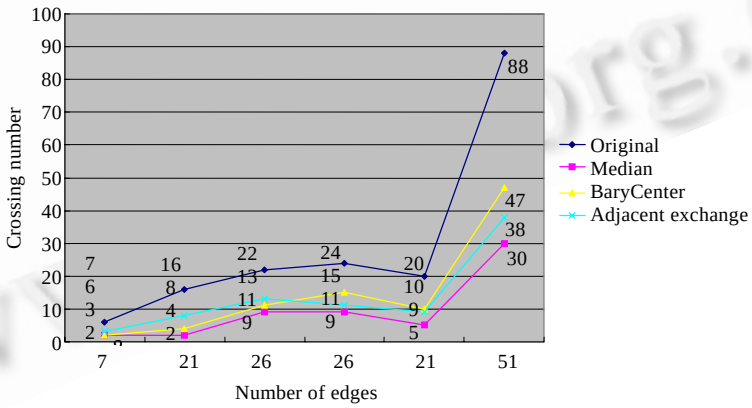


Fig.10 Comparison among crossing reduction methods

图 10 交叉最小化方法比较

Eichelberger[9]和Sugiyama[12]在复合图边交叉最小化时采用了基于交叉矩阵的方法,这种方法的优点是可以把最小化问题转化为一组矩阵运算.Eichelberger[9]在Warfield[19]的基础上推导了交换相邻节点、添加节点、删除节点等操作的矩阵表示形式,并将其用于边交叉的消解.而本文在Foster[11]交叉最小化框架下分析了UML嵌套类图的特点,并给出了基于遍历包含关系树的交叉最小化算法.它的优点是不会因算法本身而引入新的交

叉,而且在逐层消除交叉时可以方便地加入启发策略,这些都是基于交叉矩阵的方法所不具备的.在复合图交叉最小化中,交叉矩阵方法的时间复杂度为$O(V^4)$,即使采用压缩交叉矩阵进行缓存,它的时间复杂度也为$O(V^2)$,同时,还会有$O(V^2)$的空间开销[9].而文中遍历包含关系树的方法在不需要缓存中间计算结果的情况下就可以达到$O(V^2)$的时间复杂度,因此,在处理节点数目较多的复杂类图时具有明显的时空优势.

5 结 论

本文在Sun[13]的基础上选取了一组类图布局约束,分析了图元嵌套关系在层次化布图算法的各个阶段引入的问题,并对原有基于有向图层次化的UML类图布图算法在以下几个方面进行了扩展:

- 系统地分析了具有嵌套关系的类图给层次化布局算法带来的问题并给出了解决方法;
- 将包含关系、包依赖关系与图中具有层次性的边一起纳入层次分配算法进行处理;
- 通过后序遍历包含关系树来减少边边交叉和边包交叉的数目;
- 利用添加水平约束边和网络单纯形算法对横坐标分配问题进行求解.

实验结果表明,扩展的嵌套有向图布图算法能够适应逆向类图的绘制,具有层次性清晰、支持图元嵌套、边与包含节点交叉少以及总面积较小等优点.第 2.1 节中引入的包含节点依赖关系能够在一定程度上调整布局的纵横比,然而由于图中具有层次性的边组成的路径平均长度较小(平均为 9),结果图的宽度仍然较大.后续的工作将考虑在层次分配时根据层中节点的数目进一步优化调整纵横比.

References:

[1] Tilley S, Huang S. A qualitative assessment of the efficacy of UML diagrams as a form of graphical documentation in aiding program understanding. In: Proc. of the 21st Annual Int'l Conf. on Documentation. San Francisco: Association for Computing Machinery, 2003. 184−191.

[2] Gutwenger C, MichaelJünger, Klein K. A new approach for visualizing UML class diagrams. In: Proc. of the ACM Symp. on Software Visualization. San Diego: ACM Press, 2003. 179−188.

[3] Eiglsperger M, Kaufmann M, Siebenhaller M. A Topology-Shape-Metrics Approach for the Automatic Layout of UML Class Diagrams. San Diego: Association for Computing Machinery, 2003. 189−198.

[4] Seemann J. Extending the sugiyama algorithm for drawing UML class diagrams: Towards automatic layout of object-oriented software diagrams. Lecture Notes in Computer Science, 1997. 415−424.

[5] Ou SG, Liu C. Research and implementation on channel-based Hiera Chical layout algorithm. Application Research of Computers, 2004,11(59):173−174 (in Chinese with English abstract).

[6] Sun CA, Liu C, Jin MZ. Effective wove algorithm for software structure graph. Journal of Beijing University of Aeronautics and Astronautics, 2000,26(6):706−708 (in Chinese with English abstract).

[7] Tollis IG, di Battista G, Eades P, Tamassia R. Graph Drawing: Algorithms for the Visualization of Graphs. New Jersey: Prentice Hall, 1999.

[8] Huang JW, Kang LS, Chen YP. A new graph drawing algorithm for undirected graphs. Journal of Software, 2000,11(1):138−142 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/11/138.htm

[9] Holger E. Aesthetics and automatic layout of UML class diagrams [Ph.D. Thesis]. German: Würzburg University, 2005.

[10] Holger E. On class diagrams, crossings and metrics. In: Proc. of the Dagstuhl Seminars on Graph Drawing. Dagstuhl: Internationales Begegnungs-und Forschungszentrum für Informatik (IBFI), 2005. 1−12.

[11] Foster M. Apply crossing reduction strategies to layered compound graphs. In: Proc. of the 10th Int'l Symp. on Graph Drawing. Irvine: Springer-Verlag, 2002. 276−284.

[12] Sugiyama K, Kaufmann M, Siebenhaller M. Visualization of structural information: Automatic drawing of compound digraphs. IEEE Trans. on Systems, Man and Cybernetics, 1991,21(4):876−892.

[13] Sun D, Wong K. On evaluating the layout of UML class diagrams for program comprehension. In: Proc. of the IEEE Workshop on Program Comprehension. St. Louis: Institute of Electrical and Electronics Engineers Computer Society, 2005. 317−326.

[14] Gansner ER, North SC, Vo K. A technique for drawing directed graphs. IEEE Trans. on Software Engineering, 1993,19(3): 214−230.

[15] Sugiyama K, Tagawa S, Toda M. Methods for visual understanding of hierarchical system structures. IEEE Trans. on Systems, Man, and Cybernetics, 1981,11(2):109−125.

[16] Eades P, Nicholas C. Edge crossings in drawings of bipartite graphs. Algorithmica, 1994,11(4):379−403.

[17] Gansner ER, Stephen C, North KV. DAG—A program that draws directed graphs. Software-Practice and Experience, 1988,18(11): 1047−1062.

[18] Gansner ER, North SC. An open graph visualization system and its applications to software engineering. Software-Practice and Experience, 2000,30(11):1203−1233.

[19] Warfield JN. Crossing theory and hierarchy mapping. IEEE Trans. on Systems, Man and Cybernetics, 1977,7(7):505−523.

附中文参考文献:

[5] 欧胜高,刘超.一种基于通道的层次布图算法的研究和实现.计算机应用研究,2004,11(59):173−174.

[6] 孙昌爱,刘超,金茂忠.一种有效的软件结构图的布图算法.北京航空航天大学学报,2000,26(6):706−708.

[8] 黄竞伟,康立山,陈毓屏.一个新的无向图画图算法.软件学报,2000,11(1):138−142. http://www.jos.org.cn/1000-9825/11/138.htm

王晓博(1982－),男,河南平顶山人,博士生,主要研究领域为程序分析,软件可视化,软件演化分析.

王欢(1984－),女,硕士生,主要研究领域为数据挖掘,软件可视化.

刘超(1958－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件测试.