CCNeter:C程序代码Petri网自动建模工具

周国富[1,2] 孙韵秋[2] 蔡 宇[2]
(武汉大学软件工程国家重点实验室 武汉 430072)[1] (武汉大学计算机学院 武汉 430072)[2]

摘 要 CCNeter是实现扩展的Petri网——CNet自动可视化建模的工具,对程序语句从数据、操作和控制3个方面进行描述,刻画了程序代码中数据、操作以及控制之间的关系。它通过解析C工程中文件、函数模块、变量之间的依赖关系,自动形成程序的CNet规范,并根据CNet规范自动进行图形绘制和布局。CCNeter是实现程序静态分析自动化的重要前提。

关键词 Petri网,CNet,形式化技术,自动建模

CCNeter:An Automatic Modeling Tool Based on Petri Nets for C Program

ZHOU Guo-fu[1,2] SUN Yun-qiu[2] CAI Yu[2]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(School of Computer, Wuhan University, Wuhan 430072, China)[2]

Abstract CCNeter is an automatic modeling tool based on CNet, an extension of Petri nets. CCNeter respectively describes data, operations and control from a sourece code. Accordingly, on Petri nets specification the relationship among data, operation and control can be discovered. Through capturing the dependency relations among source files, functions and variables of C project, CCNeter automatically creates CNet specification for C program, then draws and lays out the specification. CCNeter is an important precondition task of static analysis of program code.

Keywords Petri nets, CNet, Formal method, Automatic modeling

1 前言

随着计算机应用不断深入社会生活,软件安全问题越来越受到重视[1]。Coverity公司实施的SCAN[2]项目(美国国土安全部资助)自2006年以来,扫描了26181个项目,其中包括280多种著名的开源系统,如Firefox,Linux,Perl,Python以及PHP等,完成分析6000多万行不重复的源码。研究表明软件漏洞问题不仅没有好转,情况依然非常严重。如此大的代码分析工作量单纯靠人工来进行,显然非常困难。人们急需一种自动化分析方法来帮助程序员发现软件中的漏洞[3-5]。

漏洞检测的传统方式是通过动态测试发现尽可能多的程序运行错误。动态测试虽然在准确率方面存在优势,但是只能在程序运行中进行,无法观察程序的内部结构和流程,同时在测试时需要额外的运行开销。早在20世纪70年代,Clarke等人[6]便提出了一种介于动态测试和程序验证之间的静态分析方法,其对代码进行解析,构建中间模型,恢复程序信息[7,8],从而分析并发现缺陷。与动态测试相比,静态分析不需要运行程序,而是应用形式化理论技术分析程序源代码[9,10]。另一方面,与程序验证相比,静态分析仅仅是发现程序中的错误,并不证明程序的完全正确性,减少了静态分析的应用难度。Basili和Selby的实验结果[4]表明,静态分析在错误发现率上可以达到动态测试的水平,并且效率远远高于动态测试。

程序的形式化是实现静态分析的重要条件。与基于逻辑、代数等描述工具相比,具有可视化特征的状态机[11]、控制流图[12]、数据流图[13]等形式描述工具得到学者和应用工程师的欢迎。然而,目前流行的可视化工具仍然缺乏有效的理论支持。Petri网[14,15]的状态和变迁二元观点可以很好地描述物理系统的动态特征,具有良好的理论基础和直观的可视化特征而广受欢迎。Petri网理论体系已经得到学术界的承认,2003年通过了ISO15909标准[16]。但是经典Petri网的绝对动态和状态观点并不完全适合程序的建模,和其他建模工具一样难以实现建模自动化。

为实现程序代码的Petri网描述,一方面提出两层程序模型,即从物理实现和计算两个角度对程序进行建模[17]。物理模型是计算的实现,由变量、读写操作、资源和控制4个物理性成分构成。计算模型则是对算法的描述,由变量、操作和控制3个计算性成分构成。两层模型通过变量和操作发生关联,程序就是物理模型和计算模型的统一。另一方面,基于两层模型,选取北京大学袁崇义教授提出的CNet[15]作为规范描述工具。本文讨论的CCNeter是实现CNet建模自动化。

CCNeter主要实现C语言程序自动转化成CNet可视化规范。在现阶段,选取C程序代码进行可视化描述是基于以

到稿日期:2010-06-08 返修日期:2010-12-20 本文受国家自然科学基金(61040036),教育部留学回国人员科研启动基金资助项目,湖北省自然科学基金(2009CDB218),中央高校基本科研专项资金(6082015),高等学校学科创新引智计划(B07037)资助。
周国富(1970—),副教授,主要研究方向为Petri网、程序测试、静态分析,E-mail:gfzhou@whu.edu.cn。

下考量:C 语言是一个用途广泛的过程性编程语言,能够把指令和数据从程序的其余部分分离出去、隐藏起来,利于抽象其中的依赖关系。同时,C 语言适用范围大,可移植性好。将来 CCNeter 会逐步增加其他编程语言支持。在文献[18]中详细介绍了从 UNITY[19] 到 CNet 的映射。UNITY 是一种并行程序描述语言,具备了一个程序语言具有的基本特征,这些研究成果是实现 CCNeter 的基本前提。同时,CCNeter 将从文件依赖、函数依赖、变量操作的依赖 3 个层次描述 C 语言程序中的依赖关系。可视化方面采用开源工具包 graphviz[20] 实现变量、操作等元素的图形生成和自动布局。可视化模型元素的交互应用开源工具包 grappa[21] 实现。

本文第 2 节详细介绍 CCNeter 基于的 Petri 网扩展元素;第 3 节介绍 CCNeter 采用的主要数据结构和算法以及调用相关 graphviz 和 grappa 工具包;最后说明我们的相关工作。

2 模型

Petri 网术语采用文献[15]的定义,在参考 Petri 网提供的基本图符语义基础上,CCNeter 采用以下图形语义。

2.1 库所(Place)

库所分为变量库所和控制库所两类。

1)控制库所(control place):当变迁发生时,需要消耗一定的托肯,同时产生一定数目的托肯。包含这种可消耗(可流动)托肯的库所是控制库所。

2)变量库所(variable place):变迁的发生读写库所中包含的内容并不消耗托肯,是一种非消耗关系。变量库所中的托肯不具流动性。

控制库所的语义和经典 Petri 网库所的语义一致。变量库所是 CNet 特有的扩展。变量库所中的托肯不具流动性,其数目是不变的。

变量库所与控制库所连接的弧类型不同。控制库所只能连接控制弧;变量库所可以连接读弧、写弧、读写弧,或者是这些弧的组合,但是不能和控制弧连接。相关类型的弧将在下面详细讨论。一般来说,变量库所描述具有数据类型的数据,如实数型、布尔型、结构等、静态属性或其他非消耗性资源。控制库所与变量库所的图形表示均采用经典 Petri 网的库所表示。

2.2 托肯(Token)

托肯除了描述资源的数目之外,也可以表示变量的当前值,此时库所中的托肯不具流动性。与经典 Petri 网一样,托肯由符号"●"表示。这里的托肯有两种语义:流动的托肯描述控制流,不可流动的托肯描述具体值。在不致混淆的情况下,不流动的托肯也可由符号"●"表示,托肯的个数就是其描述的值。同样地,可流动托肯也可由一个整数类型的值来描述,此时的值表示托肯的个数。在图 1 中,库所 Sc 的当前值是 2。为了区别于变量库所,控制库所中的托肯由图形"●"表示。

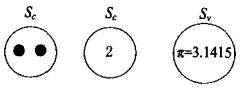

图 1 控制库所中的托肯与变量库所中的托肯

2.3 弧(Arc)

除了与经典 Petri 网弧语义一致的控制弧外,CNet 定义了读弧和写弧。读弧读取变量库所中托肯所描述的值,写弧则改变变量库所中托肯表示的当前值。CNet 有 4 种类型的弧,分别是控制弧、读弧、写弧、读写弧。

1)控制弧和经典 Petri 网中的弧语义和描述一致,类型记为 CA(见图 2(a))。控制弧描述了控制流关系,变迁的触发将消耗由控制库所包含的托肯。

2)读弧表示从变量库所中读取值,类型记为 RA(见图 2(b))。读弧语义是变迁的触发将读取由变量库所描述的变量的当前值,但不改变该变量的值。

3)写弧表示向变量库所中写入值,类型记为 WA(图 2(c))。写弧语义是变迁的触发将修改由变量库所描述变量的当前值。

4)读写弧是表示同时的读写,类型记为 RWA(见图 2(d))。读写弧是读和写的合成,其语义是变迁的触发不仅读取由变量库所描述变量的当前值,而且改变该变量库所描述变量的当前值。

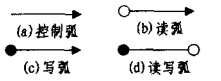

图 2 4 种弧类型

2.4 变迁(Transition)

CNet 增加哨机制控制变迁的发生。哨不仅探测资源是否满足条件,同时探测相关的变量是否满足条件。变迁的触发既可以消耗资源,也可以仅仅访问资源(读写)。变迁触发导致托肯的流动或值的访问之外,还用脚本描述变迁如何改变托肯数。

变迁由 3 部分构成,如图 3 所示。哨(Guard)有控制哨(control guard)和变量哨(variable guard)两种,分别检测关联的库所是否满足条件。如果控制库所中托肯数不满足触发条件,那么控制哨为 false,此时变迁就不能触发。当托肯数满足时,控制哨为真,记为 CG;如果变量库所中的变量值不满足条件,那么变量哨就为 false,此时变迁就不能触发。当变量条件满足时,变量哨就为真,记为 VG;变迁的发生由控制哨和变量哨共同决定。变迁至少包含控制哨和变迁哨中的一个。变迁的第 3 部分是由程序语言描述的变迁的操作语义,记为 Statement。

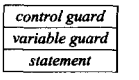

图 3 变迁的结构

下面给出一个简单例子来说明建模风格。为节省篇幅,本节直接给出例子的 CNet 规范(见图 4)。

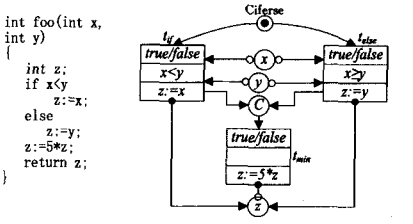

图 4 $5 * \min(x,y)$形式规范

3 实现技术

本节讨论 CCNeter 结合 graphviz, grappa 和 Jakarta-

ORO[22] 3 种开源包，实现自动建模、自动图形布局和用户交互功能。

Graphviz 是贝尔实验室研发的一个开源可视化项目。通过图布局算法，可以将图中的节点比较均匀地分布在画布上。但是 graphviz 生成的图不能进行交互。为克服这一不足，应用 grappa 作为制图与布局工具。Grappa 不仅提供了建立、控制、遍历和显示图形的节点、边和子图的方法，而且提供了创建、选择和删除图形元素等交互手段。

CCNeter 建模的基本前提是待建模的 C 程序是无语法错误的，因此不需要非常完备的语法检查和优化功能，只要抽取其中的函数、变量、操作及其之间的关系即可。本项目采取正规表达式方法实现代码关系的解析，完成程序代码中提取有价值的数据及其关系的分析处理。Jakarta-ORO 是最全面的以及优化得最好的正则表达式 API 之一。虽然 java. util. regex 和 Jakarta-ORO 均能实现字符匹配，但当字符串可能有多个子串匹配给定的正则表达式时，Jakarta-ORO 实现起来更加方便。

解析后的代码元素及其关系通过数据库保存，比放在内存中更具有效率，满足重复调用的需要。通过数据库的存储和访问，对相关表进行分割、存储冗余数据、存储衍生列、合并相关表处理，可以高效地实现查找并形成特定的数据结构。

3.1 目标

CCNeter 解析程序有 3 层依赖关系，即文件、模块、变量依赖。对 C 程序进行解析，获得文件、函数、变量、操作之间的依赖关系。把解析出的 3 层代码关系存入对应数据库或关系矩阵，再根据获得的依赖关系自动形成文件、函数、变量、操作、控制等依赖的 CNet 规范，以及这些 CNet 规范的图形自动布局，从而实现 C 语言程序代码的 Petri 网自动建模。CCNeter 可视化建模主要包含依赖关系解析、图形规范的自动生成和布局、图形界面的用户交互(见图 5)。

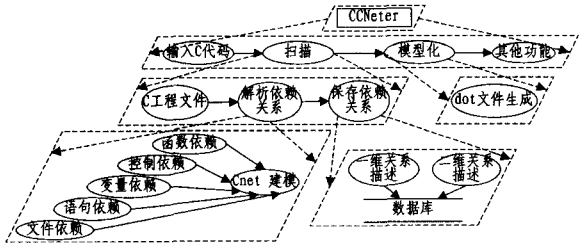

图 5 CCNeter 结构

3.2 代码关系解析

依赖性是程序模型的重要特征。CCNeter 主要给出 3 层依赖关系的实现：工程文件中文件之间的依赖关系；文件内函数之间的依赖关系；语句中变量和操作之间的关系。

为实现抽象，控制语句模块封装成一个操作。如 if 〈block〉 else 〈block〉，虽然〈block〉 中可能包含多个语句，但 CCNeter 把〈block〉作为一个语句看待，需要时 CCNeter 展开成子图。

根据第 2 节的讨论，CCNeter 只需要变迁、库所、弧 3 种基本图形元素。在程序建模中，有以下建模原则：

- 文件作为库所类型；
- 函数作为变迁类型；
- 控制语句模块作为变迁类型；
- 变量(常量)作为库所类型；
- 语句(操作)作为变迁类型；
- 关系由弧来表示。

在文件依赖层，文件作为库所，变迁则是 include，所有文件之间通过 include 发生读写关系。在函数依赖层，形参作为库所，函数作为变迁。函数对形参进行读写。在代码依赖层，变量作为库所，语句则作为变迁，变量通过语句发生读写关系。在这 3 个层次中，可能会出现计算平台通过内存、CPU 等资源来控制它们之间的关系，这些资源作为可消耗资源，用控制库所表示，相关变迁通过资源的分配发生关系。这 3 个层次并不是绝对独立的，可以互相并存。比如为了封装的需要，某些控制语句可以封装并嵌入到函数依赖层。

CCNeter 采用了两个最基本的数据结构：一个库所类型 S，一个是变迁类型 T。

库所 S 的数据结构是：

```
structure S {
int index;
//索引号，元素具有唯一的索引号；
char category;
//F：文件；V：变量；R：资源，可流动 token
bool expandable;
//封装机制，可展开，
char type;
//库所描述的变量的类型，如果：
// category=F；类型为 FILE；
// category=R；资源类型；
// category=V；变量的类型；
string statement;
//具体的字符(串)，如变量名；
string constraint;
//约束条件描述；比如：需要的内存、时间等资源；
string RA;
//被读；被变迁所读出的列表；
string WA;
//被写；被变迁所写入的列表；
string CA;
//控制流；token 向变迁流出的列表；
}
```

每个库所的索引是 16 位字符型，系统最多可以表示 65536 个不同库所。同一个文件可能被多次包含；函数可能被多次调用；这种调用和包含关系的元操作就是读/写。此时 S 元素涉及到的读变迁 RA 将由多个变迁的索引号串联构成。构成规则：把相关的索引号变成等长字符串，进行连接。比如库所 f 的索引号为 0x0011，被读 3 次，则 f 的 RA=ABEFCDEBFABC，也就是 f 被索引号 0xABEF，0xCDEB，0xFABC 的变迁所读出。WA，CA 的构成类似，只是写入和托肯的流出关系。

变迁 T 的数据结构是：

```
structure T {
int index;
//索引号，每个元素都具有唯一的索引号；
char category;
//S：语句；F：函数；B：控制语句块；
bool expandable;
//封装机制，可展开子图
```

```
string VG;
//变量哨:条件语句中的逻辑部分
string CG;
//控制哨;缺省值为 false
char type;
//变迁描述的操作的类型,如果是函数则表示函数的类型,如果是赋值语句则表示被赋值变量的类型;
int statement;
//具体语句
string constraint;
//约束条件描述;比如:需要的内存、时间等资源;
string RA;
//读;变迁所读的变量列表;
string WA;
//写;变迁所写入的变量列表;
string CA;
//控制流;token 流向的库所列表;
}
```

T类型的RA,WA,CA的串联和S类型的RA,WA,CA串联方法一致。

3.3 函数依赖

函数依赖就是一个函数的发生取决于另一个函数的发生。在C程序中以函数调用方式表现。被调用函数称为决定因素,调用函数为结果因素。本文的函数依赖借鉴这个定义。在程序的多个函数中,若存在调用关系,则抽取出函数以及与该函数有依赖关系的元素。

根据C语言的语法规则,经过一系列的预处理,写出匹配函数的正则表达式,提取出函数。函数的抽取主要采用正则表达式匹配的方法,正则表达式:

String ps1="^#(.|\n)*";

String ps2="\\[[^]]+\\]";

String ps3="(.+)\n*(\\{[^]]+\\])"。

分别的作用是:

• ps1 匹配输入的整段程序,即从文件中抽取出代码,准备进一步匹配;

• ps2 匹配出各函数块语句并存入数组,即从代码段中抽取出函数代码块;

• ps3 把各函数块分成函数名和函数体,即从函数代码块中抽取出函数名和函数体。

String ps4="\\b\\w*\\b\\(.*\\)";

String ps5="\\[[^]]+\\]";

String ps6="/*.**/|\'+\'|(\\++|\\--|\\>=|\\<=)|\\d+\\.\\d*|\\w+|\".+\"|\\S"。

分别的作用是:

• ps4 匹配输入的代码段,从中抽取出函数名,构造关联关系;

• ps5 匹配输入的代码段,从中抽取出函数代码块,准备进一步匹配;

• ps6 匹配输入的代码段,从中抽取出语句,准备进一步解析。

ps6匹配出的语句是程序中的单位语句。CCNeter将单位语句作为一个变迁,并在此基础上进一步抽取出变量,从而构成变量与单位语句之间的读写关系。

3.4 变量依赖

C程序中变量分为被读变量和被写变量两类。变量的当前值可以访问,但是并不改变其内容,称为读变量;变量的值可以访问并被改变的,称为写变量。

区分读/写变量的重要标记就是赋值符号"="。处于赋值符号左边的,其值将被改变;处于赋值符号右边的,其值一般不会变化,仅仅是读取。C语言独有的特殊运算符也可能导致赋值符号右边的变量具有写关系,如自加操作++、自减操作--、位操作等,这些操作是变量对自己的读写,如++j;。这类语句作类似于j=j+1;预处理。

函数形参的实参代入是变量的赋值。函数的返回操作也是一种赋值操作。常量是只读变量。规定常量或没有显式定义的变量是一种特殊的变量,这些变量的名字是常量的地址。如常量5可以预处理形为int add(5)的变量声明,变量名是add(5),类型是int。根据这些规则,以赋值语句为桥梁,构建出赋值符号左右边变量的读写依赖关系。

变量与语句之间的依赖关系可以根据Petri网连接矩阵定义,给出相应的读矩阵RA、写矩阵WA、控制矩阵CA。这些矩阵是程序分析的基础,将在后文中详细讨论。

3.5 模型规范生成

dot是graphviz软件包的图形描述语言,可以描述图、节点、边、标注等图元。这些图形元素可以进行动态设置。CCNeter经过解析程序代码,形成S类型元素和T类型元素以及它们之间的关系弧,只需要变迁、库所、控制弧、读弧、写弧等图形元素。S元与T元之间是RA,WA,CA关系。

由于CNet的图形元素并不是dot标准图形库中元素,需要进行特别的制定。CNet的库所由dot预定义的椭圆或者圆形表示。弧和变迁的图形规范则定义为:

• 写弧

[dir=both,arrowhead=normal,arrowtail=odot]

• 读弧

[dir=both,arrowhead=normal,arrowtail=dot]

• 读弧

[dir=both,arrowhead=odot,arrowtail=dot]

• 控制弧

[arrowhead=normal]

• 变迁

[shape=record,label=<f0>|<f1>|<f2>]

在实际建模时,根据以上图形元素模板,分别设置具体属性,从而获得不同的图形元素,用graphivz生成所需的dot文件。图6是CCNeter工具生成的$5*\min(x,y)$的模型。

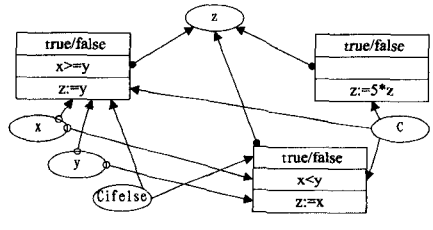

图6 CCNeter实现的$5*\min(x,y)$模型

3.6 交互性

由于基于C语言的工程涉及到多个源文件,在有限的屏幕空间无法完全显示,因此有必要对某些信息进行隐藏。采用3层模型的目的之一就是实现信息的折叠与隐藏。CC-

Neter 随时响应用户事件，与用户进行实时交互，隐藏和展开信息。

模型图中节点接收鼠标点击，如果该节点是不可再分的元素，可以弹出指定信息。否则把节点所隐藏的信息以子图方式显示出来，继续等待用户事件。CCNeter 应用 grappa 实现交互，主要接口函数 GrappaListener 接收鼠标事件，如鼠标左键双击、右键单击等。在 GrappaAdapter 类中 grappaClicked 函数处理鼠标事件。CCNeter 新建一个继承 GrappaAdapter 的子类，并在 grappaClicked 中扩展以下代码：

```
if(elem instanceof Node){
    int id=elem. getId();
      String name=elem. getName();
      JFrame newj=new JFrame(name);
      GrappaPanel lis=new GrappaPanel(subg);
          lis. setDebugGraphicsOptions(1);
      VLis vlis=new VLis();
          lis. addGrappaListener(vlis);
          newj. setContentPane(lis);
          newj. setSize(400,600);
          newj. setVisible(true);
    }
```

CCNeter 响应并弹出一个新的窗口，系统将在新窗口中绘制并显示被点击节点所隐藏的信息或者子图。CCNeter 的事件监听实现过程如下：

```
1)开始
2)在图形容器 grappaPanel 加监听器 VLis
3)loop Vlis listen 鼠标事件
4)if event == mouse clicked {
    if mouse. key == right
    Vlis. GrappaAdapter. grappaPressed()
        // 进入相关事件处理
    else if clicked == double
        Vlis. grappaClicked
        // 进入相关事件处理
  }
5)endloop
```

事件监听器机制通过在容器中添加监听器捕捉鼠标事件，大大简化了捕捉事件的过程。事件监听获得鼠标点击的图形对象的标识符。依据所获得的标识符，在后台数据库中查询相关信息，形成相应的模型 dot 描述。事实上，通过鼠标事件监听机制，可以实现其他功能，比如放大、缩小模型图。

CCNeter 继承 Grappa 中的 parser 类来处理输入 dot 格式的文件，并根据 dot 描述生成一个 graph 对象，CCNeter 调用 grappa 中的 GrappaPanel 对象容器显示这个 graph 对象，从而实现 dot 文件的绘制。

4 相关工作

传统的语义验证从问题域开始，逐步建模并验证[14,23,24]。然而在没有找到合适的自顶向下验证方法的情况下，实践更迫切需要的是从代码开始(从下至上)的验证。

本课题主要是研究面向程序缺陷的静态分析技术。之所以采用 Petri 网标准作为我们的可视化方法，是由于 Petri 网的图符元素少，却不失表达力。可执行性和图形化特征是 Petri 网受到广泛欢迎的因素之一。CNet 是 Petri 网的一种扩展，具有描述程序特征的机制。我们的工作是在形式化建模基础上，研究有效的程序缺陷分析技术，以实现静态分析的自动化。

CCNeter 的理论基础是 CNet，对程序语句从数据、操作和控制 3 个方面进行描述，对程序变量操作从读、写角度予以建模，可以有效地描述程序控制流、操作流和状态流。CCNeter 是实现 CNet 自动建模的工具，充分体现 CNet 的建模风格，并向最终用户提供可视化的使用环境，同时提供相关的算法对程序代码进行建模、分析和模拟。

参 考 文 献

[1] 梅宏，王千祥，张路，等. 软件分析技术进展[J]. 计算机学报，2009，32(9)：1697-1710

[2] SCAN[EB/OL]. http://scan. coverity. com/，May 5，2010

[3] 吴新松，周洲仪，贺也平，等. 基于静态分析的强制访问控制框架的正确性验证[J]. 计算机学报，2009，32(4)：730-739

[4] Basili V R，Selby R W. Comparing the Effectiveness of Software Testing[J]. IEEE Transactions on Software Engineering，1987，13(12)：1278-1296

[5] West B. 安全编程：代码静态分析[M]. 北京：机械工业出版社，2008

[6] Clarke L A. A System to Generate Test Data and Symbolically Execute Programs[J]. IEEE Trans. Softw. Eng.，1976，2(3)：215-222

[7] Wagner D，Dean D. Intrusion Detection via Static Analysis[C]// Proceedings of the 2001 IEEE Symposium on Security and Privacy. IEEE Computer Society，2001：156

[8] Naik M，Park C-S，Sen K，et al. Effective static deadlock detection[C]// the 31st International Conference on Software Engineering. Vancouver，Canada，2009：386-396

[9] Pousot A，Cousot R. Abstract interpretation：a unified lattice model for static analysis of programs by construction or approximation of fixpoints[C]// Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. Los Angeles，California：ACM，1977：238-252

[10] Kildall G A. A unified approach to global program optimization [C]// Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming languages. Boston，Massachusetts：ACM，1973：194-206

[11] Jet C. Bandera：Extracting finite-state models from Java source code[C]// 22nd ICSE. Limerick，Ireland，2000：439-458

[12] Liu Y S，Huang C，Xu R Z. The program control flow graph and the test path automation generation for source program[C]// 4th International Conference on Quality and Reliability(ICQR 2005). Beijing，2005：847-854

[13] Castro M，Costa M，Harris T. Securing software by enforcing data-flow integrity[C]// Proceedings of the 7th Symposium on Operating Systems Design and Implementation. Seattle，Washington：USENIX Association，2006：147-160

[14] Girault C，Valk R. Petri Nets for Systems Engineering：a guide to modeling，verification，and applications[M]. Springer-Verlag，2002

[15] 袁崇义. Petri 网理论与应用[M]. 北京：电子工业出版社. 2003

[16] Petri Nets，International Standard Ballot of ISO/IEC 15909-1 [S]. International Standard Ballot of ISO/IEC 15909-1

[17] Zhou Guo-fu，He Guo-liang. One program model for cloud com-

puting[C]// First International Conference on Cloud Computing. Lecture Notes in Computer Science, 2009: 589-594

[18] Zhou Guo-fu, Yuan Chong-yi. Mapping PUNITY to UniNet[J]. Journal of Computer Science and Technology, 2003, 18(3): 378-387

[19] Chandy K M, Misra J. Parallel program design[M]. MA, USA: Addison-Wesley Pub. Co. Inc, 1989

[20] GRAPHVIZ[EB/OL]. http://www.graphviz.org, May 5, 2010

[21] GRAPPA[EB/OL]. http://www2.research.att.com/~john/Grappa, May 5, 2010

[22] JAKARTA-ORO [EB/OL]. http://jakarta.apache.org/oro/, May 5, 2010

[23] Bjorner D, Jones C B, Airchinnigh A, et al. VDM '87 VDM—A Formal Method at Work[M]. Germany: Springer-Verlag, 1987

[24] Spivey J M. The Z Notation: A Reference Manual[EB/OL]. http://spivey.oriel.ox.ac.uk/~mike/zrm/, 1998

(上接第 95 页)

表 2 二层 SA/GA 算法的测试结果

时间段数	节点数	边数	最大值	近似度	最小值	近似度	平均值	近似度	平均 CPU 时间(s)
3	10	25	1008	1.18	980	1.15	992.9	1.16	10.57
3	10	30	1198	1.22	1123	1.15	1170.2	1.19	17.58
3	20	40	1795	1.32	1636	1.20	1728.8	1.27	36.28
3	20	50	2019	1.26	1874	1.17	1948.4	1.24	53.53
3	30	75	3345	1.33	3081	1.23	3209.4	1.29	180.63
3	30	100	4311	1.34	4107	1.28	4208.1	1.31	249.17
3	40	120	5391	1.38	5033	1.29	5288.3	1.36	376.20
3	40	140	5991	1.35	5749	1.30	5887.6	1.33	418.87
3	50	150	6720	1.39	6492	1.34	6613.4	1.37	705.36
3	50	200	8543	1.31	8249	1.27	8433.9	1.30	998.61
4	10	20	927	1.37	792	1.17	865.3	1.28	12.31
4	10	35	1454	1.34	1311	1.20	1377.1	1.26	21.70
4	20	40	1864	1.41	1678	1.27	1779.5	1.36	41.78
4	20	45	2125	1.45	1970	1.34	2031.1	1.38	61.75
4	30	60	2949	1.47	2778	1.39	2867.1	1.43	136.14
4	30	90	3843	1.43	3644	1.35	3756.0	1.39	236.41
4	40	100	4504	1.46	4240	1.37	4403.6	1.43	311.25
4	40	130	5676	1.45	5356	1.37	5529.5	1.41	538.32
4	50	170	7130	1.41	6787	1.35	6967.4	1.38	761.83
4	50	195	8306	1.42	8075	1.38	8187.9	1.40	1023.37

从表 2 可以得出，该算法可以在合理的时间给出很好的解。对于测试的实例，该算法的近似度在最坏情况下不大于 1.47，在最好情况下不大于 1.39，平均情况下不大于 1.43。另外，随着网络规模的扩大，算法的执行时间也随之增加；因为该算法是用问题下界衡量近似度，所以该算法的实际近似度要好于统计的结果。

结束语 时间依赖中国邮路问题与传统中国邮路问题相比有更广泛的应用领域，更具有普遍意义。时间依赖中国邮路问题的时变特性，使得传统问题的理论和算法不再适用。在分析时间依赖中国邮路问题的性质的基础上，提出了二层 SA/GA 算法，与已有相关算法相比，该算法能有效地解决大规模时间依赖中国邮路问题。对于具体的测试实例而言，近似度在最坏情况下不大于 1.47，实现了结果与效率平衡的目的。该算法可以应用到网络通信、实时系统测试以及智能交通系统等众多领域。

参 考 文 献

[1] 管梅谷. 奇偶图上作业法[J]. 数学学报, 1962, 1: 263-275

[2] Edmonds J, Johnson E L. Matching, Euler tours and the Chinese Postman Problem[J]. Math. Prog, 1973, 5: 88-124

[3] Papadimitriou C H. On the Complexity of Edge Traversing[J]. Journal of the ACM, 1976, 23, 544-544

[4] Frederickson G N. Approximation Algorithms for Some Postman Problem[J]. Journal of the ACM, 1979, 26, 538-554

[5] Minieka E. The Chinese postman problem for mixed networks [J]. Management Science, 1979, 253: 643-648

[6] Wm Z. On the windy postman problem on eulerian graphs[J]. Mathematical Programming, 1989, 44: 97-112

[7] Raghavachari B, Veerasarnyt J. Approximation algorithm for the asymmetryic Postman Problem[C]// Proceedings of 10th Annual ACM-SIAM Symposium on Discret Algorithms. 1999: 734-741

[8] Orloff C S. A Fundamental Problem in Vehicle Routing[J]. Networks, 1974, 4: 35-64

[9] Ghiania G, Musmannob R. A heuristic for the periodic rural postman problem[J]. Computers & Operations Research, 2005, 32: 219-228

[10] Dror M, Stern H, Trudeau P. Postman tour on a graph with precedence relation on arcs[J]. Networks, 1987, 17: 283-294

[11] Cabral E A, Gendreau M, Ghiani G, et al. Solving the hierarchical Chinese postman problem as a rural postman problem[J]. European Journal of Operational Research, 2004, 155: 44-50

[12] Korteweg P, Volgenant T. On the Hierarchical Chinese Postman Problem with linear ordered classes[J]. European Journal of Operational Research, 2006, 169: 41-52

[13] Ahr D, Reinelt G. A tabu search algorithm for the min-max k-Chinese postman problem[J]. Computers & Operations Research, 2006, 33(12): 3403-3422

[14] Daskin M C, Mark S. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms[J]. Transportation Science, 1992, 26: 185-200

[15] Malandraki C, Robert B D. A restriced dynamic programming heuristic algorithm for the time dependent traveling salesman problem[J]. European Journal of Operational Research, 1996, 90: 45-55

[16] Yang-Byung P. A solution of the bicriteria vehicle scheduling problems with time and area-dependent travel speeds[J]. Computer & Industrial Engineering, 2000, 381: 73-187

[17] Ichoua S, Gendreau M, Potvin J Y. Vehicle dispatching with time-dependent travel times[J]. European Journal of Operational Research, 2003, 144: 379-396

[18] Alberto V D, Luca M A, Norman C, et al. Time Dependent Vehicle Routing Problem with an Ant Colony System[A]// Istituto Dalle Molle di Studi sull' Intelligenza Artificiale[C]. Switzerland, IDSIA-02-03, 2003