(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103473507 A
(43)申请公布日 2013.12.25

(21)申请号 201310443173.6

(22)申请日 2013.09.25

(71)申请人 西安交通大学
地址 710049 陕西省西安市咸宁西路 28 号

(72)发明人 陶敬 周文瑜 胡文君 赵双
马小博

(74)专利代理机构 北京科亿知识产权代理事务所（普通合伙） 11350
代理人 汤东凤

(51)Int.Cl.
G06F 21/56(2013.01)

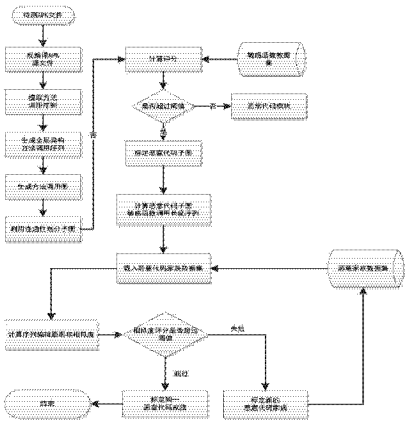
权利要求书1页 说明书6页 附图2页

(54)发明名称

一种基于方法调用图的Android恶意软件检测方法

(57)摘要

本发明提出一种Android恶意软件检测方法。采用构建Andorid应用Apk软件的异构方法调用图，标定敏感函数，利用图的连通性对恶意代码进行定位和家族分类。具体流程包括：对异构方法调用图进行图的连通性扫描，得到各个子图，对各个子图进行敏感函数打分，超过阈值的子图即为恶意代码模块，不同的Android软件中相似的恶意代码子图结构即为恶意代码家族。本发明可以启发式地发现未知恶意软件，并对其进行家族标定，为广大Android第三方市场和个人用户提供安全扫描和保护。



CN 103473507 A

1. 一种 Android 恶意代码检测方法，包括以下步骤：

第一步，采集 Android 软件的恶意样本，进行手动分析，提取其中的敏感函数；

第二步，提取所述恶意样本中经常使用的敏感接收器；

第三步，对第一步和第二步提取的敏感函数和接收器进行打分，打分原则为高危操作或者敏感信息窃取方向的分值最高，危害程度越低，分值越低；

第四步，对待测的 apk 文件，通过反编译 apk 源文件，得到内部方法调用序列，然后再提取 apk 文件对外部的敏感函数和敏感接收器进行的调用，把这两部分调用添加到内部方法调用序列中，形成全局的异构方法调用序列；

第五步，根据第四步生成的异构方法调用序列，生成异构方法调用图，图中包含外部调用敏感函数、敏感接收器、Main 函数和具体权限。

第六步，对第五步生成的图进行连通性扫描，利用图的深度遍历算法，划分出独立子图；

第七步，对上一步处理过的子图，利用第三步中的分值结构进行敏感性打分，并计算每个独立子图的评分；

第八步，上一步中评分超过第一阈值的即为恶意代码子图模块，进行标定与记录；

第九步，每检测出一个恶意代码子图后，计算该恶意代码子图的调用路径长度序列；

第十步，将该恶意代码子图的调用路径长度序列与已知恶意代码家族调用路径长度序列进行对比，并计算编辑距离，所述编辑距离是指两个调用路径长度序列之间，由一个转成另一个所需的最少编辑操作次数；

第十一步，记录每一个敏感函数的恶意代码子图的调用路径长度序列与已知恶意代码家族调用路径长度序列的编辑距离，并将该编辑距离与已知恶意代码家族调用路径长度序列中对应的敏感函数调用路径长度总和进行比值，若比值均小于第二阈值，则判定该恶意代码与所述已知恶意代码家族属于同一个家族。

2. 如权利要求 1 所述的 Android 恶意代码检测步骤，其特征在于，第一步中提取的敏感函数包括网络类、短信类、电话类、文件操作类、设备操作类、代码执行类和地理位置类七大类。

3. 如权利要求 1 所述的 Android 恶意代码检测步骤，其特征在于，第二步中，所述经常使用的敏感接收器包括接受短信接收器、接通电话接收器、挂断电话接收器、接受来电接收器和开机启动接收器。

4. 如权利要求 1 所述的 Android 恶意代码检测方法，其特征在于，第三步中的打分采用以下分值结构：分值分为六档，以 5 分为最低档，30 分为最高档，步长为 5。

5. 如权利要求 4 所述的 Android 恶意代码检测方法，其特征在于，优选的，所述第一阈值为 0.8。

6. 如权利要求 1 所述的 Android 恶意代码检测方法，其特征在于，第九步中，所述调用路径长度是指：对于每一个外部敏感函数方法，在子图中存在多个调用序列，每一个调用序列中的节点集合即为函数调用路径，而节点集合的节点数目之和即为函数调用路径长度。

7. 如权利要求 1 所述的 Android 恶意代码检测方法，其特征在于，第十步中，所述编辑操作包括将一个元素替换成另一个元素、插入一个元素和删除一个元素。

8. 如权利要求 1 所述的 Android 恶意代码检测方法，其特征在于，第十一步中，所述第二阈值为 10％。

一种基于方法调用图的 Android 恶意软件检测方法

技术领域

[0001] 本发明涉及移动互联网技术领域，主要涉及一种检测 Android 系统上的恶意代码的方法。

背景技术

[0002] 随着智能手机的高速发展，Android 平台逐渐成为了世界上第一大的移动终端平台，产品覆盖了机顶盒，手机，平板，以及各种智能终端，从各个角度影响着人们的生活。而且这些智能终端的功能越发的强大，包括了语音通话，数据业务，NFC 近场通讯等。Android 智能终端已经深入的进入到了我们每个人的生活中，支付类服务，生活类服务，地图类服务，娱乐类服务，个人信息类服务。在这样的情况下，Android 平台上的安全问题逐渐成为了一个不得不被关注的问题。

[0003] 据安全公司 TrustGo 最新数据表明，Android 上恶意应用数量自 2011 年 9 月到 2012 年 9 月增长了 580％。全球 Android 恶意软件数量已从 2 年前的百余款到了今天的过百万，从简单的窃取用户通信记录发展到了全面监控用户手机，窃取包括个人记录，银行记录，第三方软件保存信息，后台发送吸费短信，后台静默点击广告等多个方面。

[0004] 现有 Android 平台恶意代码检测技术大多采用特征库方式，对新的未知恶意软件几乎没有查杀能力。

发明内容

[0005] 针对现有技术的不足，本发明的目的在于提供一种 Android 恶意代码检测方法，通过启发式静态分析，提高对恶意代码的识别和查杀能力。

[0006] 本发明的目的是通过以下技术方案来实现的：

[0007] 一种 Android 恶意代码检测方法，包括以下步骤：

[0008] 第一步，采集 Android 软件的恶意样本，进行手动分析，提取其中的敏感函数；

[0009] 第二步，提取所述恶意样本中经常使用的敏感接收器；

[0010] 第三步，对第一步和第二步提取的敏感函数和接收器进行打分，打分原则为高危操作或者敏感信息窃取方向的分值最高，危害程度越低，分值越低；

[0011] 第四步，对待测的 apk 文件，通过反编译 apk 源文件，得到内部方法调用序列，然后再提取 apk 文件对外部的敏感函数和敏感接收器进行的调用，把这两部分调用添加到内部方法调用序列中，形成全局的异构方法调用序列；

[0012] 第五步，根据第四步生成的异构方法调用序列，生成异构方法调用图，图中包含外部调用敏感函数、敏感接收器、Main 函数和具体权限。

[0013] 第六步，对第五步生成的图进行连通性扫描，利用图的深度遍历算法，划分出独立子图；

[0014] 第七步，对上一步处理过的子图，利用第三步中的分值结构进行敏感性打分，并计算每个独立子图的评分；

[0015] 第八步，上一步中评分超过第一阈值的即为恶意代码子图模块，进行标定与记录；

[0016] 第九步，每检测出一个恶意代码子图后，计算该恶意代码子图的调用路径长度序列。

[0017] 第十步，将该恶意代码子图的调用路径长度序列与已知恶意代码家族调用路径长度序列进行对比，并计算编辑距离，所述编辑距离是指两个调用路径长度序列之间，由一个转成另一个所需的最少编辑操作次数。

[0018] 第十一步，记录每一个敏感函数的恶意代码子图的调用路径长度序列与已知恶意代码家族调用路径长度序列的编辑距离，并将该编辑距离与已知恶意代码家族调用路径长度序列中对应的敏感函数调用路径长度总和进行比值，若比值均小于第二阈值，则判定该恶意代码与所述已知恶意代码家族属于同一个家族。

[0019] 本发明的有益效果为：启发式地发现未知恶意软件，可以对其进行家族标定，提高识别和查杀能力，为广大 Android 第三方市场和个人用户提供安全扫描和保护。

附图说明

[0020] 图 1 是本发明提出的 Android 恶意代码检测方法的流程示意图；

[0021] 图 2 是计算恶意代码子图的调用路径长度序列的算法示意图。

具体实施方式

[0022] 以下结合附图对本发明的技术方案进行详细说明。

[0023] 如图 1 所示，本发明采用构建 Apk 软件异构方法调用图，标定敏感函数，继而实现以图的相关性方式对 Android 恶意代码进行定位和家族分类。在一个特定实施例中，检测方法具体包括以下步骤：

[0024] 第一步，采集 Android 软件的恶意样本，进行手动分析，提取其中的敏感函数。

[0025] 所述敏感函数包括网络类，短信类，电话类，文件操作类，设备操作类，代码执行类，地理位置类等七大类。在一个实施例中，敏感函数共计 31 个敏感 api 函数接口，如表 1 所示。

[0026] 表 1 敏感函数和敏感接收器示例

[0027]

敏感 API 分类	敏感 API 函数	敏感 API 函数评分
网络类	java/net/HttpURLConnection.disconnect	5
网络类	java/net/HttpURLConnection.getInputStream	5
网络类	java/net/URL.openConnection	20
网络类	java/net/HttpURLConnection.setRequestMethod	30
网络类	org/apache/http/impl/client/DefaultHttpClient.execute	20
网络类	org/apache/http/impl/client/HttpClient.execute	20
网络类	org/apache/http/client/methods/HttpPost.<init>	20
短信类	android/telephony/SmsManager.sendDataMessage	30
短信类	android/telephony/SmsManager.sendMultipartTextMessage	30
短信类	android/telephony/SmsManager.sendTextMessage	30
短信类	android/telephony/gsm/SmsManager.sendDataMessage	30
短信类	android/telephony/gsm/SmsManager.sendMultipartTextMessage	30
短信类	android/content/BroadcastReceiver.abortBroadcast	30

[0028]

文件操作类	java/io/FileOutputStream.write	5
文件操作类	java/io/InputStream.read	5
文件操作类	java/io/File.createNewFile	10
代码执行类	java/lang/Runtime.exec	30
代码执行类	java/lang/Process.getInputStream	5
代码执行类	java/lang/Process.destroy	20
代码执行类	java/lang/Process.getOutputStream	10
设备操作类	android/telephony/TelephonyManager.getDeviceId	5
设备操作类	android/telephony/TelephonyManager.getSubscriberId	5
设备操作类	android/telephony/TelephonyManager.getSimSerialNumber	5
设备操作类	android/content/pm/PackageManager.getApplicationInfo	5
设备操作类	android/content/ContentResolver.update	10
设备操作类	android/content/ContentResolver.delete	15
设备操作类	android/content/ContentResolver.query	15
设备操作类	android/content/ContentResolver.insert	10
地理位置类	android/telephony/gsm/GsmCellLocation.getLac	15
地理位置类	android/telephony/gsm/GsmCellLocation.getCid	15
地理位置类	android/location/Location.getLatitude	15
地理位置类	android/location/Location.getLongitude	15
电话类	android/telephony/PhoneStateListener.onCallStateChanged	20
电话类	android/media/MediaRecorder.start	30
敏感接收器	android.intent.action.BOOT_COMPLETED	30
敏感接收器	android.provider.Telephony.SMS_RECEIVED	30
敏感接收器	android.intent.action.NEW_OUTGOING_CALL	30
敏感接收器	android.intent.action.NEW_OUTGOING_CALL_FROM_CALLMASTER	30
敏感接收器	android.intent.action.PHONE_STATE	30

[0029] 第二步，提取恶意样本中经常使用的敏感接收器。本领域技术人员应当明了，所谓“经常使用”是指在预定时间内使用次数达到或超过预定阈值。在一个实施例中，经常使用的敏感接收器包括接受短信接收器，接通电话接收器，挂断电话接收器，接受来电接收器，开机启动接收器共计五个，见表 1。

[0030] 第三步，对第一步和第二步采集的敏感函数和接收器进行打分。分值可以采用连续的数值，也可以采用分段档位。在一个实施例中，分值分为六档，如表 1 所示，以 5 分为最低档，30 分为最高档，步长为 5。打分原则为高危操作或者敏感信息窃取方向的分值最高，随着危害程度不同，危害越低，分值越低。

[0031] 第四步，对待测的 apk 文件，通过反编译 apk 源文件，得到内部方法调用序列，然后再提取 apk 文件对对外部的敏感函数的调用，敏感接收器调用，由于这两部分调用，结构是

内部函数调用外部函数的形式，所以把这两部分调用，添加到内部方法调用序列中，从而形成了全局的异构方法调用序列。

[0032] 第五步，根据第四步生成的异构方法调用序列，生成异构方法调用图，其中，图中包含外部调用敏感函数，敏感接收器，Main 函数。

[0033] 第六步，对第五步生成的图进行连通性扫描，利用图的深度遍历算法，划分出独立子图。

[0034] 第七步，对上一步处理过的子图，利用附录一所示分值进行敏感性打分，主要利用第三步设计的分值结构，并计算每个独立子图的评分，（评分 = 敏感分数 / 方法总数）。

[0035] 第八步，上一步中评分超过阈值的即为恶意代码子图模块，进行标定与记录，阈值为 0.8；

[0036] 第九步，每检测出一个恶意代码子图后，计算恶意代码子图的调用路径长度序列。函数调用路径长度计算方法：在子图中，对每一个外部敏感函数方法，必然存在多个调用序列，每一个调用序列中的节点集合即为函数调用路径，而节点集合的节点数目之和即为函数调用路径长度。每一个外部敏感函数都有不止一个调用序列，这些不同的调用序列所生成的调用路径长度所构成的集合即为函数调用路径长度序列。

[0037] 参考图 2，上述算法的示例可描述如下：

[0038] 图中 In 表示图中每个节点的入度，Seq 则记录每个节点的调用路径长度。

[0039] Step1：首先对全图所扫描，计算每个节点的入度 (In)，并初始化调用路径长度序列 Seq。

[0040] Step2：首先选取图中入度为零的节点 A，并查找 A 节点所调用的所有节点：B 和 C，将 A 中的 Seq 中的每个元素加 1 合并到 B 中，并且给 B 的 Seq 中直接再加一个元素 1（代表了 A 直接调用 B），得到 B：Seq[1]。最后将 B 节点的入度减一，对对等的 C 节点做同样操作。

[0041] Step3：继续选取图中入读为零的节点 B，并查找 B 节点所调用的所有节点 D，将 B 的 Seq 中的每个元素加 1 合并到 D 的 Seq 中，得到 D：Seq[2]，再将 D 的 Seq 中直接再加一个元素 1（代表了 B 直接调用 D）得到 D：Seq[1,2]，将 D 节点入度减一。

[0042] Step4：继续选取图中入读为零的节点 C，并查找 C 节点所调用的所有节点 D、E，将 C 中的 Seq 的每个元素加 1 合并到 D 的 Seq 中，得到 D：Seq[1,2,2]，再将 C 的 Seq 中直接再加一个元素 1（代表了 C 直接调用 D）得到 D：Seq[1,1,2,2]，将 D 节点入度减一。对 E 做类似操作，得到了 E：Seq[1,2]，E 的入度减一。

[0043] Step5：继续选取图中入读为零的节点 D，并查找 D 节点所调用的所有节点 E，将 D 中的 Seq 的每个元素加 1 合并到 E 的 Seq 中，得到 E：Seq[1,2,2,2,3,3]，再将 D 的 Seq 中直接再加一个元素 1（代表了 D 直接调用 E）得到 E：Seq[1,1,2,2,2,3,3]。

[0044] Step6：全图遍历结束，得到了 E 节点的调用路径长度序列 E：Seq[1,1,2,2,2,3,3]。

[0045] 第十步，将恶意代码子图的调用路径长度序列与已知恶意代码家族调用路径长度序列进行对比，计算编辑距离。这里将编辑距离定义进行了扩充，指两个调用路径长度序列之间，由一个转成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个元素替换成另一个元素，插入一个元素，删除一个元素。设序列 a，b 为待测序列，i，j 分别代表 a，b 序列的指定位元素。其中，a_i 表示 a 序列第 i 个元素，b_j 表示 b 序列第 j 个元素，$distance_{a,}$

$_b$(i，j) 为序列之间的对应长度的编辑距离：

[0046] $$distance_{a,b}(i,j)=\begin{cases}\sum_{k=0}^{j} b_k, i=0\\ \sum_{k=0}^{i} a_k, j=0\\ \min\begin{cases}distance_{a,b}(i-1,j)+a_i\\ distance_{a,b}(i,j-1)+b_j\\ distance_{a,b}(i-1,j-1)+|a_i-b_j|\end{cases}\end{cases}$$

[0047] 第十一步，记录每一个敏感函数的恶意代码子图的调用路径长度序列与已知恶意代码家族调用路径长度序列的编辑距离，并将其与已知恶意代码家族调用路径长度序列中对应的敏感函数调用路径长度总和进行比值，若比值均小于 10％时，则可以判定这两个恶意代码属于同一个家族。

[0048] 由此，本发明利用图的连通性对恶意代码进行定位和家族分类，启发式地发现未知恶意软件，可以对其进行家族标定，提高识别和查杀能力。

[0049] 以上利用具体个例对本发明的原理及实施方式进行了阐述，以上实施例的说明只是用于帮助理解本发明的方法及其核心思想；同时，对于本领域技术人员，依据本发明的思想，在具体实施方式及应用范围上均会有改变之处，综上所述，本说明书内容不应理解为对本发明的限制。

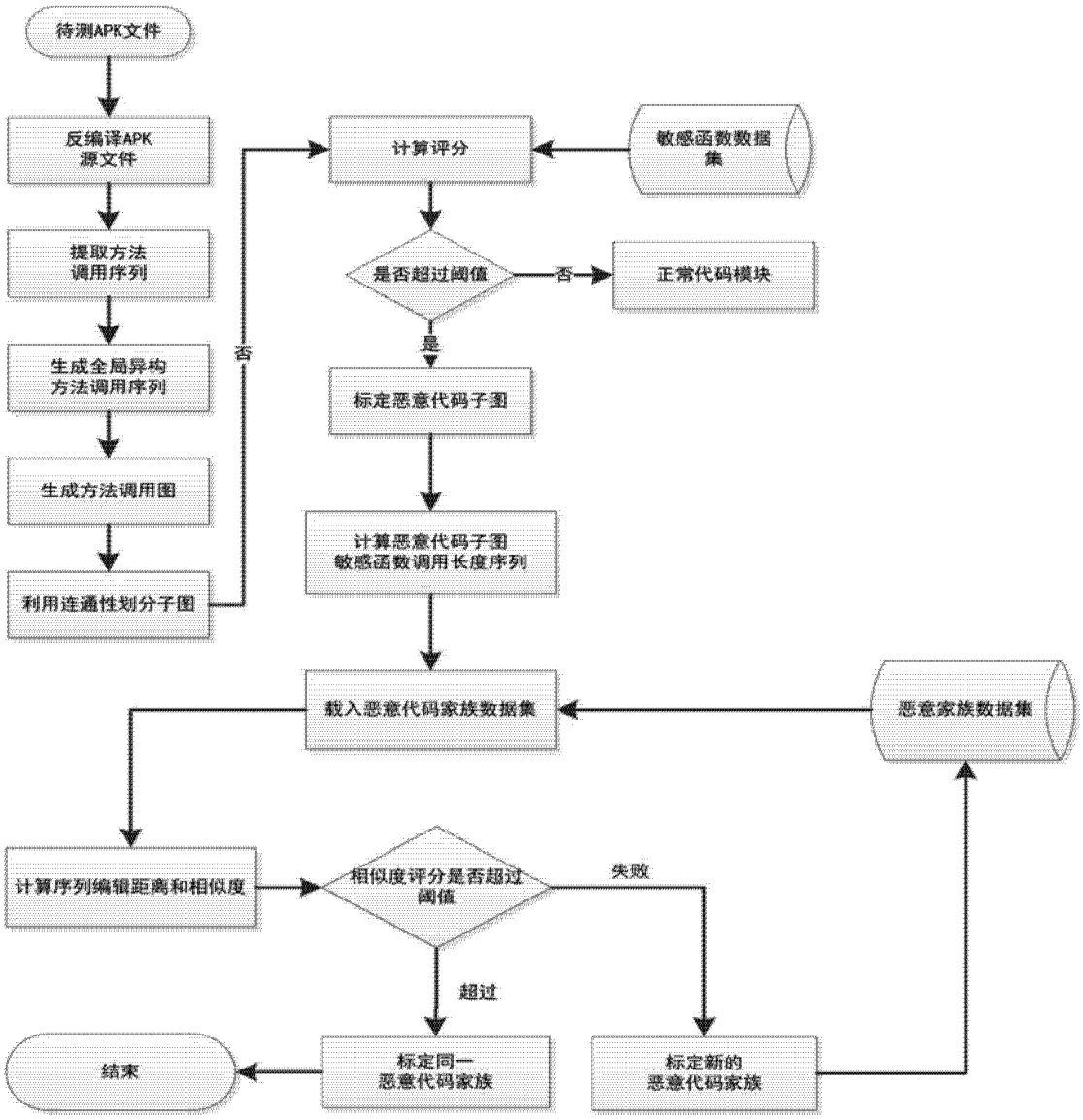

待测APK文件
反编译APK源文件
提取方法调用序列
生成全局异构方法调用序列
生成方法调用图
利用连通性划分子图
否
计算评分
敏感函数数据集
是否超过阈值
否
正常代码模块
是
标定恶意代码子图
计算恶意代码子图敏感函数调用长度序列
载入恶意代码家族数据集
恶意家族数据集
计算序列编辑距离和相似度
相似度评分是否超过阈值
失败
超过
结束
标定同一恶意代码家族
标定新的恶意代码家族

图 1

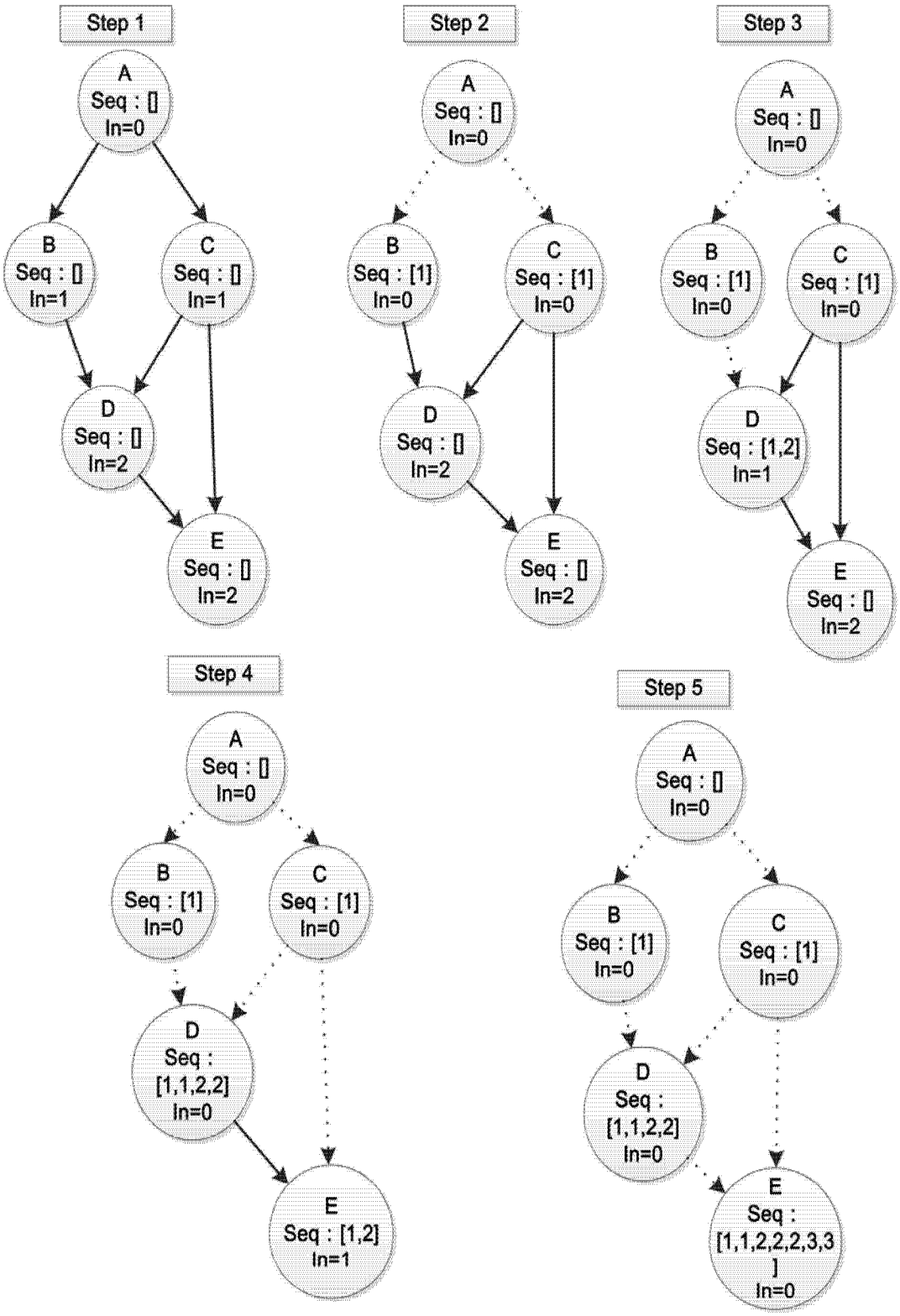

Step 1
A
Seq : []
In=0
B
Seq : []
In=1
C
Seq : []
In=1
D
Seq : []
In=2
E
Seq : []
In=2
Step 2
A
Seq : []
In=0
B
Seq : [1]
In=0
C
Seq : [1]
In=0
D
Seq : []
In=2
E
Seq : []
In=2
Step 3
A
Seq : []
In=0
B
Seq : [1]
In=0
C
Seq : [1]
In=0
D
Seq : [1,2]
In=1
E
Seq : []
In=2
Step 4
A
Seq : []
In=0
B
Seq : [1]
In=0
C
Seq : [1]
In=0
D
Seq :
[1,1,2,2]
In=0
E
Seq : [1,2]
In=1
Step 5
A
Seq : []
In=0
B
Seq : [1]
In=0
C
Seq : [1]
In=0
D
Seq :
[1,1,2,2]
In=0
E
Seq :
[1,1,2,2,2,3,3
]
In=0

图 2