计算机科学
COMPUTER SCIENCE

# 基于迭代轨迹划分的单分支循环程序终止性分析

王垚, 李轶

引用本文

王垚, 李轶. 基于迭代轨迹划分的单分支循环程序终止性分析[J]. 计算机科学, 2023, 50(9): 108-116.

WANG Yao, LI Yi. Termination Analysis of Single Path Loop Programs Based on Iterative Trajectory Division [J]. Computer Science, 2023, 50(9): 108-116.

---

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220700214

# 基于迭代轨迹划分的单分支循环程序终止性分析

王 垚[1,2] 李 轶[1]
1 中国科学院重庆绿色智能技术研究院自动推理与认知重庆市重点实验室 重庆 400714
2 中国科学院大学重庆学院 重庆 400714
(wangyao@cigit.ac.cn)

**摘 要** 秩函数作为循环程序终止性分析的重要方法已得到广泛研究。文中着重研究了单分支循环的终止性。首先提出了双向迭代循环概念,将单分支循环分为双向迭代循环和非双向迭代循环。其次,针对双向迭代循环程序,建立了一种划分思路,提出了三段式秩函数的概念,并证明了若该双向迭代循环存在三段式秩函数,则其是终止的。而对非双向迭代循环,引用增函数的划分思路,即利用增函数将原程序空间划分为更小的空间,并通过计算更小空间上的秩函数来证明原程序的终止性。最后,将三段式秩函数的计算问题归结为 SVM 分类问题,并利用工具 Z3 或 bottema 对由 SVM 所得的候选秩函数进行验证。

**关键词**:程序验证;秩函数;机器学习;程序终止性
**中图法分类号** TP311

# Termination Analysis of Single Path Loop Programs Based on Iterative Trajectory Division

WANG Yao[1,2] and LI Yi[1]
1 Chongqing Key Laboratory of Automated Reasoning and Cognition,CIGIT,CAS,Chongqing 400714,China
2 Chongqing School,University of Chinese Academy of Sciences,Chongqing 400714,China

**Abstract** The ranking function has been extensively studied as an important method of program termination analysis. In this paper,we focus on the termination of single-path loops. Firstly,the concept of two-way iterative loops is proposed,and the single-path loops are divided into bidirectional iterative loops and non-bidirectional iterative loops. Secondly,for the bidirectional iterative loop program,a division method and concept of trivial ranking function are proposed,and it is proved that if a bidirectional iterative loop exists a trivial rank function,it is terminated. As for the non-bidirectional iterative loop,we use incremental function as the division method,i.e.,the original program space is divided into smaller spaces by using incremental function,and the termination of the original program is proved by computing the ranking function on the smaller space. Finally,the problem of computing the trivial ranking function comes down to the SVM classification problem,and we verifies candidate ran-king functions using the tools Z3 or bottema.

**Keywords** Program verification,Ranking functions,Machine learning,Program termination

## 1 引言

随着计算机相关技术的发展,人类目前正处于大数据和人工智能的时代,相关软件对社会的影响也越来越深入,计算机已经成为人们生活中必不可缺的东西。在人们越来越依赖计算机软件的同时,需要对软件内部程序的稳定性与正确性加以关注。众所周知,计算机程序是整个计算机服务体系的核心,因此,为了避免程序故障导致的重大事故,有必要研究程序运行的正确性与可靠性,保证程序的稳定运行。

在程序理论的范畴中,程序的正确性指程序在一定的前提条件下,经过预先设定好的运算,生成符合某些性质的结果,同时要求程序最后必须终止。研究程序的终止性是保障程序正确性的必要条件。虽然循环程序的终止性问题已经被证明为不可判定问题,但是仍可以通过一定的算法判断循环程序的终止性。

根据判断程序是否终止可以将其分成两类方法:通过寻找秩函数判断程序终止和通过寻找不动点或不变集来判断程序不终止。文献[1-14]的工作都是针对线性循环求解线性

到稿日期:2022-07-24 返修日期:2022-11-30
基金项目:国家自然科学基金(61572024,11771421,61103110);重庆市自然科学基金(cstc2019jcyj-msxmX0638);重庆市院士专项(cstc2018jcyj-yszxX0002,cstc2019yszx-jcyjX0003);国家重点研发计划(2020YFA07123000);中国科学院"西部之光"
This work was supported by the National Natural Science Foundation of China(61572024,11771421,61103110),National Science Foundation of Chongqing,China(cstc2019jcyj-msxmX0638),Chongqing Science and Technology Innovation Guidance Special Project(cstc2018jcyj-yszxX0002,cstc2019yszx-jcyjX0003),National Key Research and Development Program of China(2020YFA07123000) and Chinese Academy of Sciences "Light of West China" program.
通信作者:李轶(zm_liyi@163.com)

秩函数，其中 Podelski 等[3]基于 Farkas 引理，根据循环条件和赋值语句构造约束关系来求解线性秩函数；Bradley 等[4]首先提出了合成线性字典序秩函数的概念，基于这个概念，Ben-Amram 等[6]提出了合成线性字典序秩函数的完备算法；自此，秩函数的形式被拓宽，Bagnara 等[9]提出了最终线性秩函数的概念，利用划分思想提高获得线性秩函数的可能性；Leike 等[12]首次提出了多阶段秩函数和嵌套秩函数的定义；Ben-Amram 等[14]证明针对单分支线性约束程序，上述两个概念是等价的，并给出了合成嵌套秩函数的完备方法。文献[15-21]针对非线性秩函数，其中 Cousot[15]使用参数抽象、拉格朗日松弛以及半正定规划进行约束求解，获得非线性秩函数；Chen 等[16]将多项式秩函数问题规约到半代数系统问题的求解中，并通过柱形代数分解的方法来获得非线性秩函数。但是这两种方法都存在一定的问题，Cousot 的方法基于 SDP 进行求解，由于 SDP 内部采用浮点计算，因此最终得到的秩函数候补不是精确结果，且没有进行验证，Chen 等的方法基于传统柱形代数分解，在最糟糕的情况，时间复杂度甚至会达到双指数。

近年来，机器学习方法被逐渐运用于程序验证领域，并且对以上方案进行了很好的改进。Yuan 等[19]将秩函数的两个满足条件转化为二分类问题，利用 SVM 算法寻找秩函数；Sun[20]基于机器学习算法，得到了多阶段秩函数的求解算法。但文献[19]的工作仅局限于全局秩函数，对拥有多阶段秩函数的循环不能很好地处理；而文献[20]的方法在进行多阶段秩函数的求解前，必须设定最大阶数，但是在理论上关于多阶段秩函数存在的阶数上界没有得到证明。

本文主要考虑了单分支线性约束循环的终止性问题。Bagnara 等在文献[9]中首次提出了增函数的概念，并利用增函数对原始程序空间进行划分得到更小的空间，进而在新的空间上计算秩函数。增函数划分为程序终止性分析提供了新的思路，即能否通过某种划分方法，将原始程序对应的程序空间划分为不同的 $m$ 个子空间，然后通过证明 $m$ 个子空间都有秩函数来证明原始程序是可终止的。受此启发，我们观察到一类被称为双向迭代循环的程序，其程序空间可以分为 3 个子空间，并证明了如果这 3 个子空间都存在秩函数，那么该程序是终止的。然后，通过 SVM 算法去探测这 3 个子空间的秩函数。针对非双向迭代循环，我们引用了文献[9]中增函数的划分思路，即利用增函数将原程序空间划分为更小的空间，并通过计算更小空间上的秩函数来证明原程序的终止性。但与文献[9]利用 farkas 引理计算秩函数的方法不同的是，本文将子空间上的秩函数计算归结为二分类问题，并利用 SVM 对其求解。不同于文献[14,19]，本文方法可以处理非线性循环的秩函数计算问题，且对部分程序，本文方法可以得到形式更为简洁的秩函数。

本文第 2 章介绍程序终止性判断的相关概念；第 3 章提出双向迭代循环以及三段式秩函数的定义，证明了若一个双向迭代循环存在三段式秩函数，则该双向循环程序终止，并给出了整个算法流程；第 4 章展示本文方法与已有方法的实验对比；最后总结全文并展望未来。

## 2　准备知识

本章将主要介绍有关程序终止性判断的相关理论基础，这些内容将在后面的算法中起到重要作用。

本文工作主要聚焦于以下形式的循环：

while $x\in\Omega$, do $x'=F(x)$

$\Omega\triangleq x\in R^n: P_1(x)\geqslant 0,\cdots,P_k(x)\geqslant 0$

其中，$\Omega$ 表示该循环的程序空间，并且 $\Omega$ 是一个单连通集，$P_i(x)$是系数为实数的多项式，$F(x)=(f_1(x),\cdots,f_m(x))$，$f_i(x)$ 是属于 $R[x]$上的多项式。在下文中，为方便阐述，我们有时用程序空间 $\Omega$ 来刻画所对应的循环程序。为了方便读者对于后续概念的理解，我们用下列循环贯穿全文进行阐述。

Example1：

while$(x+y>0)$, do $x'=x-1, y'=-2y$

### 2.1　不变集

**定义 1**(不变集)　给定循环程序 $\Omega$，集合 $Q\subseteq\Omega$ 被称为程序 $\Omega$ 的不变集，如果 $\forall x\in Q$，有$x'\in Q$。

由不变集的定义可知，若循环程序存在不变集，那么不变集中的点将在此集合内进行无穷迭代，无法跳出该集合，表明该循环程序无法终止。

### 2.2　秩函数

秩函数法作为当前终止性分析的主流方法已得到广泛研究。给定一个循环程序，若能找到它的秩函数，则表明该循环是终止的。

**定义 2**(秩函数)　给定循环程序 $\Omega$，函数 $f(x)$被称为该程序的秩函数，如果它满足以下条件：

$$\forall x\in\Omega, f(x)\geqslant c', c'>0 \tag{1}$$

$$\forall x\in\Omega, f(x)-f(x')\geqslant c' \tag{2}$$

$x'$为 $x$ 迭代一次获得的状态点，上述秩函数的定义与传统秩函数定义是等价的。传统秩函数的定义如下：

$$\forall x\in\Omega, f(x)\geqslant 0 \tag{3}$$

$$\forall x\in\Omega, f(x)-f(x')\geqslant c \tag{4}$$

为了方便实验，本文在秩函数的定义上采用定义 2 中的式(1)、式(2)，而传统定义中的式(3)、式(4)可以等价改写为式(1)、式(2)。这是因为，如果函数 $f(x)$ 满足定义 2 中的式(3)、式(4)，则在式(3)两端同时加上一个正数 $c$ 且令 $f'(x)=f(x)+c$，$(c>0)$，则有 $f'(x)\geqslant c$；同时不难看出，$f'(x)-f'(x')=f(x)+c-(f(x')+c)=f(x)-f(x')\geqslant c$，显然$f'(x)$满足式(1)、式(2)。反之，若存在 $f(x)$使得式(1)、式(2)成立，则令 $c'=c$，那么式(3)、式(4)平凡成立。

由式(1)、式(2)的性质可知，借助秩函数有界和递减的两个性质，将循环程序的状态映射到一个良基集中，程序每经过一次循环迭代，秩函数的值都会减少，直到抵达秩函数下界。由此可得，如果循环程序存在秩函数，那么它一定能够终止。

### 2.3　增函数与最终线性秩函数

下面介绍文献[9]中增函数的概念，该概念将被应用于下文对非双向迭代循环的终止性分析中。

**定义 3**(增函数)　给定循环程序 $\Omega$，函数 $q(x)$被称为 $\Omega$ 上的增函数，如果对任意的 $x\in\Omega$,，有 $q(x')-q(x)\geqslant 1$，其中 $x'$为 $x$ 迭代一次获得的状态点。

若增函数是线性的，那么它刻画了一个超平面。在增函数的划分下，循环空间中的状态点经过有限次迭代后都会进入划分超平面上方与循环空间相交的部分。接着，在该相交

的部分中寻找线性秩函数。

而在被划分得到的子空间寻找到的秩函数被称为最终线性秩函数[9]。增函数的引入，将寻找原先整个循环空间上秩函数的问题转化为寻找循环空间子空间上秩函数的问题，增大了线性秩函数被找到的可能性。

Example2：

while$(x\geqslant 0)$, do $x'\leqslant x+y, y'\leqslant y-1$

对于上述循环，其增函数为 $f(x,y)=-y$，因为 $f(x',y')-f(x,y)=-(y-1)-(-y)=1$。

## 3 单分支循环程序终止性判定

本章将主要研究单分支循环的终止性问题。增函数划分法的引入，将在整个循环空间寻找秩函数的问题转化为在其子空间寻找，这增大了秩函数被找到的可能性。受此启发，本文的一个基本思路是：能否通过某种不同的划分方法，将原先的循环空间划分为多个不相交的子空间，并通过证明每个子空间都存在秩函数，进而证明循环程序是终止的。为此我们根据循环空间中状态点的迭代路径特点，提出了一种新的循环程序类型，即双向迭代循环。这类循环的程序空间可被超平面分为 3 个部分，更重要的是，寻找这 3 个子空间中的秩函数即可判断整个循环程序的终止性。因此，这类循环终止性问题的判断符合提出的基本思路。此外，对于非双向迭代循环，本文仍然引用文献[9]中增函数的划分方法对程序空间进行划分。在寻找这两类循环被划分后所得的子空间中的秩函数时，我们证明其子空间中秩函数的计算问题都可以归结于二分类问题，故可利用 SVM 算法获得秩函数。

### 3.1 基于迭代路径的划分

#### 3.1.1 双向迭代循环

循环空间中状态点的迭代由赋值语句决定，其迭代路径可能为沿着某一固定方向前进，也有可能在整个循环空间中进行没有规律的变化。在各种迭代路径中，我们发现，有一类循环的迭代路径会沿着某个超平面来回跃迁。下文中，我们称这类循环为双向迭代循环，并给出定义。

**定义 4**（双向迭代循环） 给定一个循环程序 $P$：while $x\in\Omega$, do $x'=F(x)$，如果存在超平面 $g(x)=\boldsymbol{a}^{\mathrm{T}}x+\boldsymbol{b}=0$，使得下列条件成立时，则 $P$ 被称为双向迭代循环。

$$\begin{cases}\Omega_1=\Omega\cap\{x\in R^n:g(x)>0\}\neq\varnothing\\ \Omega_2=\Omega\cap\{x\in R^n:g(x)<0\}\neq\varnothing\end{cases}\tag{5}$$

$$\begin{cases}\forall x.(x\in\Omega_1\Rightarrow g(x')\leqslant 0)\\ \forall x.(x\in\Omega_2\Rightarrow g(x')\geqslant 0)\end{cases}\tag{6}$$

由该定义可知，超平面 $g(x)=0$ 将循环空间 $\Omega$ 分为 3 部分：$\Omega_1,\Omega_2,\Omega_0=\Omega\cap\{x\in R^n:g(x)=0\}$。$\Omega_1,\Omega_2$ 非空，所以 $\Omega_1$ 中的状态点迭代一次后跳入超平面中或超平面下方；同样，$\Omega_2$ 中的状态点迭代一次后跳入超平面中或超平面上方。

下面的定理 1 表明，对任给的双向迭代循环，若状态点 $x$ 在 $\Omega_0$ 中，那么该状态点迭代后得到的下一个状态点 $x'$ 依旧在该超平面上，即 $g(x')=0$。

**定理 1** 给定一个循环程序 $P$：while $x\in\Omega$, do $x'=F(x)$。若 $P$ 为双向迭代循环，那么：$\forall x\in\Omega_0$，则有 $x'=F(x)\in\{x\in R^n:g(x)=0\}$。

证明：

由循环 $P$ 的定义可知，$\Omega$ 是单连通集，且 $\Omega_1\neq\varnothing,\Omega_2\neq\varnothing$，所以 $\Omega_0$ 不为空。假设存在 $x^*\in\Omega_0$ 使得 $g(F(x^*))\neq 0$，即 $x^*$ 迭代一次后所得的 $F(x^*)$ 不落在 $g(x)=0$ 上。那么由 $g(x)$，$F(x)$ 的连续性可知，必存在 $O^o(\delta,x^*)$ 邻域，使得情形 A 或 B 发生：

情形 A $F(O^o(\delta,x^*))\subseteq\{x\in R^n:g(x)<0\}$

情形 B $F(O^o(\delta,x^*))\subseteq\{x\in R^n:g(x)>0\}$

为了不失一般性，假设情形 A 发生，则必存在 $y^*\in O^o(\delta,x^*)\cap\Omega_2$，有 $g(F(y^*))<0$，也即 $y^*$ 在迭代一次后仍然在 $g(x)<0\}$ 中。这显然与循环程序 $P$ 为“双向迭代循环”的定义相悖，因为根据定义，$\Omega_2$ 中的点 $y^*$ 迭代一次后将跳入 $\{x\in R^n:g(x)\geqslant 0\}$。同理，对情形 B 也可以用如上方法证明。

综上所述，若 $P$ 为双向迭代循环，那么：$\forall x\in\Omega\cap\{x\in R^n:g(x)=0\}$，有 $x'=F(x)\in\{x\in R^n:g(x)=0\}$。□

定理 1 表明，若循环程序 $P$ 为双向迭代循环，那么任意 $\Omega_0$ 中的点 $x$，迭代一次后获得的状态点始终在 $g(x)=0$ 这个超平面上，即：$\forall x\in\Omega_0\Rightarrow g(F(x))=0$。因此有以下情况出现：$\Omega_0$ 中的点 $x$ 迭代一次后获得的新迭代点 $x'$，有可能落在 $\Omega_0$ 或 $\{x\in R^n:g(x)=0\}\backslash\Omega_0$ 中。不难看出，若对 $\forall x\in\Omega_0$，都有 $x'=F(x)\in\Omega_0$，则 $\Omega_0$ 为该循环的不变集。显然，若 $\Omega_0$ 为不变集，则循环程序 $P$ 是不终止的，这为双向迭代循环 $P$ 是不终止的提供了充分的判准。

接下来将介绍如何判断一个循环程序 $P$ 是否是双向迭代循环，根据定义 4 可知，这等价于寻找一个满足式(5)和式(6)的划分超平面 $g(x)=0$。

因此，可以通过式(5)和式(6)，利用 redlog 等工具去计算双向迭代循环的划分超平面。

Example3：针对 Example1，考虑该循环的划分超平面为 $g(x)=ax+by+c$，用该式替换定义 4 中的式(5)和式(6)中的 $g(x)$，通过 redlog 命令求解，消掉式(5)和式(6)中的 $x$ 和 $y$，获得关于 $a,b,c$ 的约束关系，发现当 $a=0,b=1,c=0$ 时，满足 $a,b,c$ 的约束关系，获得该循环的划分超平面为 $y=0$。

如果循环程序 $P$ 是双向迭代循环，接下来将证明若 $\Omega_1$，$\Omega_2,\Omega_0$ 上存在秩函数 $f_1(x),f_2(x),f_0(x)$，那么循环程序 $P$ 就是终止的，接下来给出三段式秩函数的概念。

#### 3.1.2 三段式秩函数

**定义 5**（三段式秩函数） 给定一个双向迭代循环 $P$，我们称函数 $f(x)$ 是 $P$ 的三段式秩函数，如果它满足式(8)—式(10)：

$$f(x)=\begin{cases}f_1(x)=\boldsymbol{a}_1^{\mathrm{T}}U_1(x),\forall x\in\Omega_1\\ f_2(x)=\boldsymbol{a}_2^{\mathrm{T}}U_2(x),\forall x\in\Omega_2\\ f_0(x)=\boldsymbol{a}_0^{\mathrm{T}}U_0(x),\forall x\in\Omega_0\end{cases}\tag{7}$$

$$\forall x\in\Omega_1\Rightarrow\begin{cases}f_1(x)\geqslant c, & c>0\\ f_1(x)-f_1(x'')\geqslant c, & x''=F^2(x)\end{cases}\tag{8}$$

$$\forall x\in\Omega_2\Rightarrow\begin{cases}f_2(x)\geqslant c, & c>0\\ f_2(x)-f_2(x'')\geqslant c, & x''=F^2(x)\end{cases}\tag{9}$$

$$\forall x\in\Omega_0\Rightarrow\begin{cases}f_0(x)\geqslant c & ,c>0\\ f_0(x)-f_0(x')\geqslant c, & x'=F(x)\end{cases}\tag{10}$$

其中，$U_i(x)=(x^\alpha:|\alpha|\leqslant d)$，$x^\alpha=x_1^{\alpha_1}\cdots x_n^{\alpha_n}$，$|\alpha|=\alpha_1+\alpha_2+\cdots\alpha_n$，比如，当 $d=2,n=2$，$U_i(x)=(x_1^0x_2^0,x_1^0x_2^1,x_1^0x_2^2,x_1^1x_2^0,x_1^1x_2^1,x_1^1x_2^2,x_1^2x_2^0,x_1^2x_2^1,x_1^2x_2^2)$。

由定义 5 可知，满足式(8)－式(10)的函数 $f_1(x)$，$f_2(x)$，$f_0(x)$分别为$\Omega_1$，$\Omega_2$，$\Omega_0$上的秩函数。

接下来证明，对于给定的双向迭代循环 $P$，若上述定义的三段式秩函数存在，则该双向迭代循环 $P$ 终止。

**定理 2**　给定一个双向迭代循环程序 $P$，若 $P$ 存在满足上述定义的三段式秩函数，那么 $P$ 终止。

证明：假设 $P$ 不终止，那么 $\Omega$ 上存在无穷迭代序列 $S=\{x_i|_{i=0}^{+\infty}\}$。由 $P$ 是双向迭代循环可知，该无穷迭代序列存在 3 种情形：

情形 1　$S$ 中的点在$\Omega_1$与$\Omega_2$上相互交替迭代，即$\{x_{2k}\}\subseteq\Omega_1$且$\{x_{2k+1}\}\subseteq\Omega_2$；或$\{x_{2k}\}\subseteq\Omega_2$且$\{x_{2k+1}\}\subseteq\Omega_1$；

情形 2　$S$ 中的点只出现在$\Omega_0$上，即 $S\subseteq\Omega_0$；

情形 3　$S$ 中的初始点$x_0$从$\Omega_1$或$\Omega_2$开始，即$x_0\in\Omega_1$(或$x_0\in\Omega_2$)经过有限次迭代后跳到$\Omega_0$上，由定理 1 可知，之后其始终在$\Omega_0$中迭代。

对于情形 3 可以发现，虽然初始点$x_0\in\Omega_1$(或$x_0\in\Omega_2$)，但有限次迭代后都会落入$\Omega_0$中。由于 $S$ 是无穷迭代序列，舍去情形 3 中未进入$\Omega_0$中的部分状态点，所得到的新的序列仍然是无穷迭代序列，因此情形 3 可以归为情形 2。在接下来的证明中，我们只需要讨论情形 1 和情形 2 即可。

下面分别证明当情形 1 和情形 2 发生时，$P$ 不终止的假设不成立。

1)考虑情形 1，假设 $S$ 在$\Omega_1$与$\Omega_2$上相互交替，不失一般性，下文考虑$\{x_{2k}\}\subseteq\Omega_1$且$\{x_{2k+1}\}\subseteq\Omega_2$这种情况，那么会出现以下序列：

(1)$\Omega_1$：$x_0$，$x_2$，$x_4$，…

(2)$\Omega_2$：$x_1$，$x_3$，$x_5$，…

由三段式秩函数的定义可知，$f_1(x)$和$f_2(x)$是$\Omega_1$与$\Omega_2$上的秩函数，所以有：

$$f_1(x_{2i})-f_1(x_{2i+2})\geqslant c$$

$$f_2(x_{2i+1})-f_1(x_{2i+3})\geqslant c$$

继而有：

$$f_1(x_0)>f_1(x_2)>f_1(x_4)>\cdots$$

$$f_2(x_1)>f_2(x_3)>f_2(x_5)>\cdots$$

又因为 $S$ 为无穷迭代序列，所以存在$x_n$，$x'_n$，使得$f_1(x_n)<c$且$f_2(x'_n)<c$，与 $f(x)$的定义$f_1(x)\geqslant c$且$f_2(x)\geqslant c$相悖，因此情形 1 不存在。

2)考虑情形 2，假设 S 只出现在$\Omega_0$上，$f_0(x)$是$\Omega_0$上的秩函数，那么对于序列 $S$ 则有：

$$f_0(x_i)-f_0(x_{i+1})\geqslant c$$

继而有：

$$f_0(x_0)>f_0(x_1)>f_0(x_2)>\cdots$$

又因为 $S$ 为无穷迭代序列，所以存在$x_n$，使得$f_0(x_n)<c$与 $f(x)$的定义$f_0(x)\geqslant c$相悖，因此情形 2 不存在。

综上，若双向迭代循环存在三段式秩函数，那么该双向迭代循环一定终止。□

在判断双向迭代循环终止性时，我们先判断划分超平面与循环空间 $\Omega$ 相交的部分，即$\Omega_0$是否为不变集。如果$\Omega_0$是不变集，那么双向迭代循环不终止；如果$\Omega_0$不是不变集，那么我们寻找该双向迭代循环是否具有三段式秩函数。

### 3.2　计算秩函数

对于双向迭代循环，我们利用满足定义 4 的超平面进行划分，得到两个互不相交的子空间；对于非双向迭代循环，我们引用文献[9]中增函数的方法进行划分，得到子空间。无论使用哪种划分方法，划分后都会将原空间中秩函数的计算问题归结于子空间中秩函数的寻找问题。

不失一般性，令划分后的子空间为$\Omega_i$，该空间上的秩函数为 $f_i(x)$：

$$f_i(x)=\boldsymbol{a}_i^{\mathrm{T}}U_i(x),x\in\Omega_i \tag{11}$$

由于$f_i(x)$是$\Omega_i$上的秩函数，故有

$$\forall(x,x_m),x\in\Omega_i\wedge x_m=F^m(x)\Rightarrow\boldsymbol{a}_i^{\mathrm{T}}U_i(x)\geqslant c\wedge\boldsymbol{a}_i^{\mathrm{T}}(U_i(x)-U_i(x_m))\geqslant c \tag{12}$$

令

$$G_{i,1}(x,x_m)=U_i(x) \tag{13}$$

$$G_{i,2}(x,x_m)=U_i(x)-U_i(x_m) \tag{14}$$

$\Omega$ 中不同区域上的点按$G_{i,j,j=1,2}$映射到象空间，从而得到 $\Omega$ 中各区域在该映射下的象集：

$$G_{i,1}(\Omega_i)=\{u|u=U_i(x),x\in\Omega_i\wedge x_m=F^m(x)\} \tag{15}$$

$$G_{i,2}(\Omega_i)=\{u|u=U_i(x)-U_i(x_m),x\in\Omega_i\wedge x_m=F^m(x)\} \tag{16}$$

继而可以得到：

$$\begin{cases}\boldsymbol{a}_i^{\mathrm{T}}\cdot G_{i,1}(\Omega_i)\geqslant c\\ \boldsymbol{a}_i^{\mathrm{T}}\cdot G_{i,2}(\Omega_i)\geqslant c\end{cases} \tag{17}$$

令 $S=G_{i,1}(\Omega_i)\cup G_{i,2}(\Omega_i)$。综上所述，对于任意循环程序，其子空间$\Omega_i$存在秩函数等价于存在$\boldsymbol{a}_i^{\mathrm{T}}$和正数 $c$ 使式(17)成立。

文献[20]将寻找嵌套秩函数的问题转化为寻找某个超平面的问题，该超平面能够将某个集合与原点严格分离。此证明思路可以被应用于证明定理 3。

**定理 3**　记号同上。给定一个循环程序 $P$，令其划分后的子空间为$\Omega_i$，若其子空间$\Omega_i$上存在秩函数，当且仅当存在一个超平面 $L$ 能够将集合 $S$ 上的点和原点 $\boldsymbol{o}$ 严格分离。

证明：

1)假设给定的循环程子空间$\Omega_i$存在秩函数，即：

$$f_i(x)=\boldsymbol{a}_i^{\mathrm{T}}U_i(x),x\in\Omega_i$$

我们需要证明存在超平面 $L$ 能够将集合 $S$ 上的点和原点 $\boldsymbol{o}$ 严格分离。

令 $\boldsymbol{w}=\boldsymbol{a}_i^T$，构造超平面 $L=\boldsymbol{w}\cdot\boldsymbol{u}-\frac{1}{2}c$，因为 $o=(0,0,0,0,\cdots,0)^{\mathrm{T}}$，并且 $c>0$，所以 $L(o)=\boldsymbol{w}\cdot\boldsymbol{o}-\frac{1}{2}c=-\frac{1}{2}c<0$。

对于$\forall\boldsymbol{u}\in S$，$L(u)=\boldsymbol{w}\cdot\boldsymbol{u}-\frac{1}{2}c\geqslant c-\frac{1}{2}c=\frac{1}{2}c>0$，因此超平面 $L$ 能够将集合 $S$ 上的点和原点 $\boldsymbol{o}$ 严格分离。

2)假设存在超平面 $L=\boldsymbol{w}\cdot\boldsymbol{u}+\boldsymbol{b}$ 能够将集合 $S$ 上的点和原点 $\boldsymbol{o}$ 严格分离，接下来证明程序$\Omega_i$上的确有秩函数，这等价于证明存在$\boldsymbol{a}_i^{\mathrm{T}}$使得秩函数成立。

存在两种情况：

$$(1)\begin{cases}L(o)<0\\ L(\boldsymbol{u})>0,\forall\boldsymbol{u})\in S\end{cases}$$

$$(2)\begin{cases}L(o)>0\\ L(\boldsymbol{u}))<0,\forall\boldsymbol{u})\in S\end{cases}$$

本文只讨论情况(1),类似的方法可应用于情况(2)。

$L(o)<0$,可以推出 $\boldsymbol{w}\cdot\boldsymbol{o}+b<0$,即 $b<0$,$-b>0$。

又因为 $L(\boldsymbol{u})>0$,$\forall \boldsymbol{u}\cdot S$,即 $\boldsymbol{w}\cdot\boldsymbol{u}+b>0$,$\boldsymbol{w}\cdot\boldsymbol{u}>b$,$-\frac{w}{b}\cdot\boldsymbol{u}>1$,令 $\boldsymbol{w}'=-\frac{\boldsymbol{w}}{b}$,可得,对于 $\forall \boldsymbol{u}\cdot S$,$\boldsymbol{w}'\cdot\boldsymbol{u}>1$。令 $c=1$,那么对于 $\forall \boldsymbol{u}\cdot S$,则有:

$$\begin{cases}\dfrac{\boldsymbol{a}_i^{\mathrm{T}}}{-b}\cdot G_{i,1}(\Omega_i)\geqslant c\\ \dfrac{\boldsymbol{a}_i^{\mathrm{T}}}{-b}\cdot G_{i,2}(\Omega_i)\geqslant c\end{cases}$$

因此存在 $\frac{\boldsymbol{a}_i^{\mathrm{T}}}{-b}$ 和 $c=1$ 使得式(17)成立,故该循环程序子空间 $\Omega_i$ 上的秩函数为:

$$f_i(x)=\frac{\boldsymbol{a}_i^{\mathrm{T}}}{-b}U_i(x),x\in\Omega_i$$

综上,给定一个循环程序 $\Omega$,其子空间 $\Omega_i$ 上存在秩函数,当且仅当存在一个超平面 $L$ 能够将集合 $S$ 上的点和原点 $\boldsymbol{o}$ 严格分离。□

### 3.3 验证

由于通过 SVM 算法获得的秩函数为秩函数候补,因此还需进一步验证其是否在对应的空间上满足有界和下降这两个条件。本文实验使用工具 bottema 验证候选秩函数,其他工具如 Z3 也可以进行验证。为了使候选秩函数有更大的概率通过验证,我们利用迭代的思想缩小原先的验证空间,获得新的待验证区间,接着在此区域上对候选秩函数进行验证,具体思路如下。

对于一个循环程序 $P$,假设 $f_k(x)$ 为其划分后子空间 $\Omega_k$ 上的候选秩函数。利用缩小区域法验证 $f_k(x)$ 的思路如下:首先在 $\Omega_k$ 上验证 $f_k(x)$ 是否满足有界和下降两个条件,若满足,则 $f_k(x)$ 是 $\Omega_k$ 上的秩函数,若不满足,则令 $\Omega_k^{(1)}=\Omega_k\cap F(\Omega)$,其中 $F(\Omega)=\{x\in\Omega:F(x)\in\Omega\}$,获得缩小后的子空间 $\Omega_k^{(1)}$,$\Omega_k^{(1)}$ 为 $\Omega_k$ 的子集;接着在 $\Omega_k^{(1)}$ 验证 $f_k(x)$ 是否为满足条件的秩函数,若满足,则 $f_k(x)$ 是 $\Omega_k^{(1)}$ 上的秩函数,若不满足,则继续调用缩小区域算法进行验证,令 $\Omega_k^{(2)}=\Omega_k^{(1)}\cap F^2(\Omega)$,$\Omega_k^{(2)}$ 表示 $\Omega_k$ 经过两次缩小区域法获得的子空间,以此类推直到达到缩小次数上界。

下面定理 4 将证明,虽然 $f_k(x)$ 为其子空间 $\Omega_k$ 上的候选秩函数,但是只要 $f_k(x)$ 是 $\Omega_k$ 经过 $i$ 次缩小区域法获得的某个子空间 $\Omega_k^{(i)}$ 上的秩函数,则 $\Omega_k$ 中的点最终都会跳出 $\Omega_k$。

**定理 4**(记号同上) 令 $\Omega_k^{(i)}$ 为对 $\Omega_k$ 调用 $i$ 次算法获得的子空间。若 $f_k(x)$ 是 $\Omega_k^{(i)}$ 上的秩函数,则 $\Omega_k$ 中的点最终都会跳出 $\Omega_k$。

证明:由已知可得,$\Omega_k^{(i)}=\Omega_k^{(i-1)}\cap F^i(\Omega)$,为子空间 $\Omega_k^{(i-1)}$ 调用一次缩小区域算法获得,将 $\Omega_k^{(i-1)}$ 分为两部分:$\Omega_k^{(i)}$ 和 $\Omega_k^{(i-1)}\backslash\Omega_k^{(i)}$,令 $\bar{\Omega}_k^{(i)}=\Omega_k^{(i-1)}\backslash\Omega_k^{(i)}$。因为 $f_k(x)$ 是 $\Omega_k^{(i)}$ 上的秩函数,所以 $\Omega_k^{(i)}$ 中的点经过有限次迭代将会跳出 $\Omega_k^{(i)}$,那么就会有以下两种情况发生:$\Omega_k^{(i)}$ 中的点直接跳出 $\Omega_k$ 或者落入 $\Omega\backslash\Omega_k^{(i)}$,$\Omega\backslash\Omega_k^{(i)}=\bar{\Omega}_k\cup\bar{\Omega}_k^{(1)}\cup\cdots\bar{\Omega}_k^{(i)}$(其中 $\bar{\Omega}_k=\Omega_k\backslash\Omega_k^{(1)}$,$\cdots$,$\bar{\Omega}_k^{(i)}=\Omega_k^{(i-1)}\backslash\Omega_k^{(i)}$),根据缩小区域法的缩小规则可知,$\Omega_k^{(1)}$ 排除了 $\Omega_k$ 中迭代一次就会跳出 $\Omega$ 的状态点,即 $\bar{\Omega}_k$ 中的点迭代一次就会跳出 $\Omega$,所以 $\bar{\Omega}_k^{(i)}$ 是迭代 $i$ 次就会跳出 $\Omega$ 的状态点的集合。由此可知,$\Omega\backslash\Omega_k^{(i)}$ 中的状态点经过有限次迭代后都会跳出循环空间 $\Omega$,因此,$\Omega_k^{(i)}$ 中的点经过有限次迭代后都会跳出循环空间 $\Omega_k$。综上可得,若 $f_k(x)$ 是 $\Omega_k^{(i)}$ 上的秩函数,$\Omega_k$ 中的点最终都会跳出 $\Omega_k$。□

由定理 4 可知,若 $\Omega_k$ 经过缩小区域法得到的子空间中存在秩函数,那么 $\Omega_k$ 中的点最终都会跳出 $\Omega_k$,因此我们接下来将证明整个程序 $P$ 终止。

**定理 5**(记号同上) 如果 $\Omega_k$ 中的点最终都会跳出 $\Omega_k$,那么程序 $P$ 终止。

证明:

下面将分别对非双向迭代循环和双向迭代循环进行讨论:对于非双向迭代循环,经过 $i$ 次缩小区域算法后获得 $\Omega_k^{(i)}$,该区域中的点经过有限次迭代跳出 $\Omega_k$ 的迭代路径如图 1 所示。

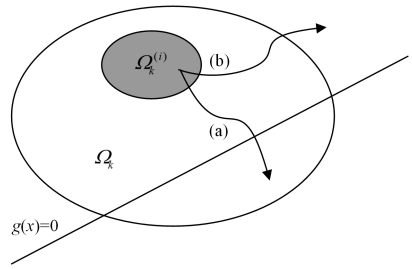


图 1 非双向迭代循环状态点迭代路径

Fig. 1 Non bi-directional loop state point iteration path

路径(a)表示 $\Omega_k^{(i)}$ 中的点跳入区域 $\Omega\backslash\Omega_k$,由于非双向迭代使用增函数划分,这表明 $\Omega$ 中的点进入 $\Omega_k$ 不会再进入区域 $\Omega\backslash\Omega_k$,所以路径(a)不成立;那么仅剩路径(b),该路径表示,$\Omega_k^{(i)}$ 中的点跳出循环空间。因此,对于非双向循环,$\Omega_k$ 中的点最终都会跳出循环空间。故非双向迭代循环程序终止。

对于双向迭代循环,为了不失一般性,我们考虑 $\Omega_1$ 经过 $i$ 次缩小区域算法后,获得区域的 $\Omega_1^{(i)}$。该区域中的点经过有限次迭代跳出 $\Omega_1$ 的迭代路径如图 2 所示。

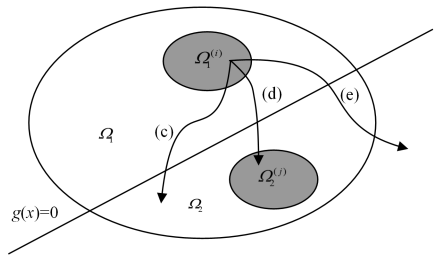


图 2 双向迭代循环状态点迭代路径

Fig. 2 Bi-directional loop state point iteration path

路径(c)表示 $\Omega_1^{(i)}$ 中的点跳入区域 $\Omega_2\backslash\Omega_2^{(j)}$,而该区域中的点经过有限次迭代都会跳出循环空间;路径(d)表示 $\Omega_1^{(i)}$ 中的点跳入区域 $\Omega_2^{(j)}$,对于这条路径,我们考虑定理 2 的证明方法,可以证明不会存在交替于 $\Omega_1^{(i)}$ 与 $\Omega_2^{(j)}$ 间的无穷迭代序列,所以这条路径上的状态点最终也会跳出循环空间;路径(e)表示 $\Omega_1^{(i)}$ 中的点跳出循环空间。所以对于双向循环,$\Omega_1$ 中的点最终都会跳出循环空间。同理,$\Omega_2$ 中的点最终都会跳出循环空间,故双向迭代循环程序终止。综上,如果 $\Omega_k$ 中的点最终都会跳出 $\Omega_k$,那么程序 $P$ 终止。

### 3.4　程序所需的子算法

本节将对基于迭代轨迹划分方案中所需要的部分重要子算法进行介绍，以便读者理解方案的具体执行细节。

#### 3.4.1　判断循环程序是否存在不变集

通过寻找输入集合的不变集来判断循环程序是否不终止，若存在不变集，则判定结果为存在不变集，否则为不存在不变集。对双向迭代循环的判定算法如算法 1 所示，非双向迭代循环的判定算法如算法 2 所示。

**算法 1**　RSTW() recurrent set of twoway loop

输入：循环程序 Ω，迭代映射 F，划分 $f_{tw}$

输出："true" or "false"

1. if $x\in(R^n:f_{tw}(x)=0\}\cap\Omega)\wedge x'=F(x)\Rightarrow x'\in(R^n:f_{tw}=0\}\cap\Omega)$ then
2.　return false;
3. else
4.　return true;
5. end if

**算法 2**　RSUTW() recurrent set of un-twoway loop

输入：循环程序 Ω，迭代映射 F，增函数 $f_{inc}$

输出："true" or "false"

1. if $x\in(x\in f_{inc}(x)\geqslant 0\cap\Omega)\wedge x'=F(x)\Rightarrow x'\in(f_{inc}(x)\geqslant 0\cap\Omega)$ then
2.　return false;
3. else
4.　return true;
5. end if

#### 3.4.2　验证候选秩函数

通过输入经过划分后获得的子空间，以及每个子空间上的候选秩函数，验证候选秩函数是否准确，若通过验证，则认为获得了准确的秩函数，否则调用缩小区域法对验证空间进行缩小，在新获得的验证区域上继续进行验证，若调用缩小区域算法的次数超过最大设定值，则认为候选秩函数未通过验证。对双向迭代循环的判定算法如算法 3 所示，非双向迭代循环的判定算法如算法 4 所示。

**算法 3**　CTW() certificate twoway loop

输入：被 $f_{tw}$ 划分后的到的 $\Omega_1,\Omega_2,\Omega_0$，秩函数 $f_1,f_2,f_0(x)$ 可调参数 $c(c>0)$，调用次数 $t_1=0$，最大调用次数 t

输出："true" or "false"

1. if $t_1>t$
2.　return false;
3.　if$(x\in\Omega_1\Rightarrow(f_1(x)>0$ and $f_1(x)-f_1(x'')>c))$ and$(x\in\Omega_2\Rightarrow(f_2(x)>0$ and $f_2(x)-f_2(x'')>c))$ and$(x\in\Omega_0\Rightarrow(f_0(x)>0$ and $f_0(x)-f_0(x')>c))$ then
4.　return false;
5.　else
6.　$t_1=t_1+1$;
7.　narrow domain;
8.　CTW();
9.　end if
10. end if

**算法 4**　CUTW() certificate un-twoway loop

输入：被 $f_{inc}$ 划分后的到的 $\Omega_1$，秩函数 $f_1$，可调参数 $c(c>0)$，调用次数 $t_1=0$，最大调用次数 t

输出："true" or "false"

1. if $t_1>t$ then
2.　return false;
3. end if
4. if$(x\in\Omega_1\Rightarrow(f_1(x)>0$ and $f_1(x)-f_1(x')>c))$ then
5.　return false;
6. else
7.　$t_1=t_1+1$;
8.　narrow domain;
9.　CTW();
10. end if

#### 3.4.3　判断双向迭代循环是否终止

判断双向迭代循环是否终止，首先通过寻找不变集来判断循环程序是否不终止，接着通过寻找秩函数来判断循环程序是否终止，具体步骤如算法 5 所示。

**算法 5**　JTW()judge twoway loop

输入：循环程序 Ω，迭代映射 F

输出："true" or "false"

1. 寻找循环的划分 $f_{tw}$；
2. if RSTW()then
3.　print"循环程序不终止"；
4.　return true;
5. else
6.　SVM();
7.　if CTW() then
8.　print"循环程序终止"；
9.　return true;
10.　else
11.　return false;
12.　end if
13. end if

#### 3.4.4　判断非双向迭代循环是否终止

判断非双向迭代循环是否终止和判断双向迭代循环是否终止的思路大致相同，区别在于前者执行算法前需要提前设置最大划分次数，具体步骤如算法 6 所示。

**算法 6**　JUTW() judge un-twoway loop

输入：循环程序 Ω，迭代映射 F，寻找次数 $p_1=0$，最大寻找次数 p

输出："true" or "false"

1. while $p_1<p$
2.　$p_1=p_1+1$;
3.　寻找循环的划分 $f_{inc}$；
4.　if RSTW()then
5.　print"循环程序不终止"；
6.　return true;
7.　else
8.　SVM();
9.　if CUTW() then
10.　print"循环程序终止"；
11.　return true;
12.　else
13.　return false;
14.　end if
15.　end if
16. end while
17. return false.

#### 3.4.5　判断循环程序是否终止

划分方案的完整执行流程如算法 7 所示。首先寻找输

入循环程序的不动点，再分别进行双向迭代循环以及非双向迭代循环的判断，如果超过了非双向迭代判定算法中设置的最大划分次数跳出后，则认为基于迭代轨迹的划分方案无法对输入的循环程序进行处理，输出结果“循环程序未知”。

**算法 7** RSTW() recurrent set of twoway loop

输入：循环程序 Ω，迭代映射 F

输出：“循环程序终止”或者“循环程序不终止”或者“循环程序未知”

```
1. if FP() then
2.     return“循环程序不终止”；
3. else
4.     if !JTW() then
5.         if !JUTW() then
6.             print“循环程序未知”；
7.             return true；
8.         else
9.             break；
10.        end if
11.    else
12.        break；
13.    end if
14. end if
```

### 3.5 流程图

整个算法的运行流程如图 3 所示。

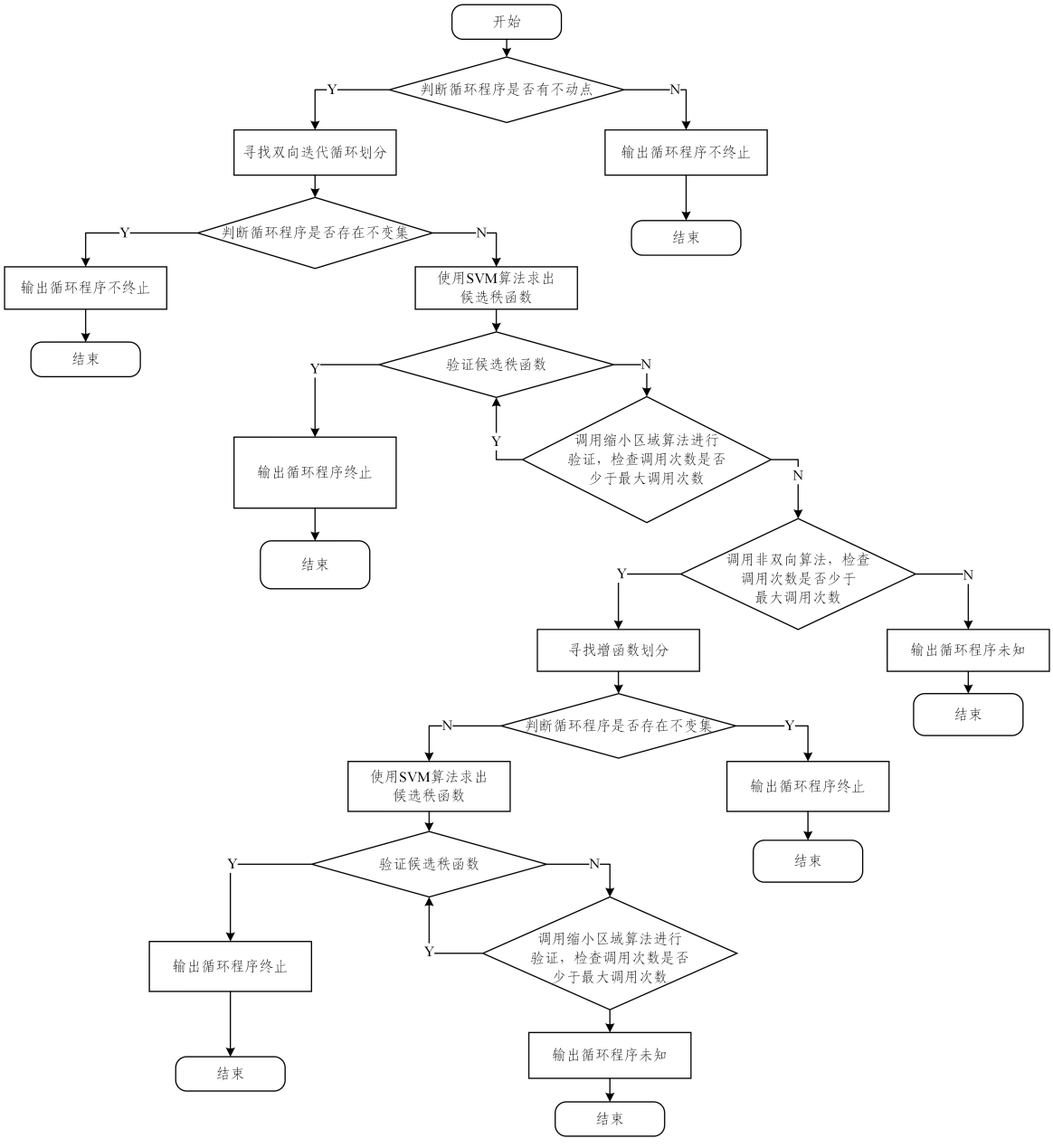


图 3 算法流程图

Fig. 3 Flow chart of algorithm

## 4 实验结果

将寻找秩函数问题转化为 SVM 分类问题，能够处理文献[14]不能处理的非线性循环。同时，本文方法利用区域划分的思想，在更小的区域上寻找秩函数，提高了找到秩函数的可能性。本实验使用 Python 3.8 进行编译，硬件信息为：

IntelCore i5-9400F@2.90GHz CPU，Nvidia GTX 1660 Ti 显卡，8GB 内存。

具体的实验用例与结果如表 1 所列。表 1 中终止性一栏中符号“F”代表循环不终止，符号“T”代表循环终止；后三列表示不同方法对该行例子的实验结果，符号“×”代表不能判断循环程序的终止性，符号“√”代表能够正确判断其终止性，此处本文方法和文献[14]算法获得的是线性模板的秩函数。对于表 1 中的例子，文献[19]均不能找到线性模板的秩函数，所以获得的是次数不大于 3 的非线性模板的秩函数。

表 1　实验结果

Table 1　Experiment results

| 序号 | 类别 | 循环体 | 终止性 | 本文方法 | 文献[14] | 文献[19] |
|---|---|---|---|---|---|---|
| 1 | 非双向 | while($x\leqslant -1$) do $x'=x+y,y'=y-1$ | F | √ | √ | × |
| 2 | 非双向 | while ($x\leqslant 4$) do $x'=x-y,y'=x+y$ | F | √ | √ | × |
| 3 | 非双向 | while ($x\geqslant 1$) do $x'=x+y,y'=y$ | F | √ | √ | × |
| 4 | 双向 | while ($x\geqslant 1$) do $x'=x+y,y=-2y$ | F | √ | √ | × |
| 5 | 双向 | while ($x+1\leqslant y$) do $x'=x+y,y'=-2y$ | F | √ | √ | × |
| 6 | 双向 | while ($x\leqslant 9$) do $x'=-y,y'=y+1$ | F | √ | √ | × |
| 7 | 非双向 | while ($x>0$) do $x'=x+y,y'=y-1$ | T | √ | √ | √ |
| 8 | 非双向 | while ($x>0$) do $x'=x-2y,y'=y+1$ | T | √ | √ | √ |
| 9 | 非双向 | while ($x>0,y<0$) do $x'=x+y,y'=y-1$ | T | √ | √ | √ |
| 10 | 非双向 | while ($x+y\geqslant 0$) do $x'=x+y,y'=y-1$ | T | √ | √ | √ |
| 11 | 非双向 | while ($x\geqslant 0$) do $x'=x+y-1,y'=\frac{3}{5}y-\frac{4}{5}$ | T | √ | √ | √ |
| 12 | 非双向 | while ($x\geqslant 4$) do $x'=x-y,y'=x+y$ | T | √ | √ | × |
| 13 | 双向 | while ($y\geqslant 0,2x+y\geqslant 0$) do $y'=10-x,x'=x-1$ | T | √ | √ | × |
| 14 | 双向 | while ($x-y>0$) do $x'=-x+y,y'=4x$ | T | √ | √ | × |
| 15 | 双向 | while ($x\geqslant 1$) do $x'=x+y,y'=-y-1$ | T | √ | √ | √ |
| 16 | 双向 | while ($x>0$) do $x'=y,y'=y-1$ | T | √ | √ | × |
| 17 | 双向 | while ($x>0$) do $x'=x+y-5,y'=-2y$ | T | √ | √ | × |
| 18 | 双向 | while ($x+y>0$) do $x'=x-1,y'=-2y$ | T | √ | √ | × |
| 19 | 双向 | while ($y^2-y\leqslant x$) do $x'=x+y-5,y'=-y$ | T | √ | × | √ |
| 20 | 双向 | while ($x^2-x+1\leqslant y^2$) do $x'=x^2+y+1,y'=-y+1$ | T | √ | × | √ |
| 21 | 非双向 | while ($x>0$) do $x'=x+y,y'=y+z$ | F | √ | √ | × |
| 22 | 非双向 | while ($x+y\geqslant 0,x<z$) do $x'=2x+y,y'=y+1,z'=z$ | T | √ | √ | × |
| 23 | 双向 | while ($x\geqslant 0$) do $x'=x+y,y'=z,z'=-z-1$ | T | √ | √ | × |
| 24 | 非双向 | while ($x>0,y>-1,z^2>1,x>y$) do $x'=x-2y,y'=y+1,z'=\frac{1}{z}y$ | T | √ | × | √ |
| 25 | 非双向 | while ($y\leqslant x,y<1,z>1$) do $x'=4\ y^2-4y+3,y'=y-2,z'=\frac{1}{2}z-1$ | T | √ | × | √ |
| 26 | 非双向 | while ($x+y\geqslant 0,y<1,z+n>0,n>1$) do $x'=x+y,y'=y-1,z'=z,n'=n-1$ | T | √ | × | √ |

表 1 中例 1－2，4－9，15－18，21－23 来源于文献[14]；例 19－20 来源于文献[19]；例 3，10－14 来源于文献[22]。

本文方法主要与文献[14]的多阶段秩函数和文献[19]的利用 SVM 寻找秩函数的方法作对比。其中，相比文献[14]，本文的工作使用 SVM 算法求解秩函数，在此过程中需要进行采样，这就使得本文工作的时间代价自然比文献[14]求解线性秩函数的方法高，但是本文方法可以处理非线性循环，并且可以构造非线性秩函数模板，增大了秩函数被寻找到的可能性，而文献[14]的方法只能求解线性循环程序的线性秩函数。

表 2 选取了表 1 中本文和文献[19]的方法都能处理的例子，序号栏中的序号与表 1 一致。其中$t_1$栏代表本文方法所用的验证时间，$t_2$栏代表文献[19]方法所用的验证时间。

表 2　验证时间对比

Table 2　Comparison of verification time

| 序号 | 本文模板 | 文献[19]模板 | $t_1$/ms | $t_2$/ms | $\Delta t=t_2-t_1$/ms |
|---|---|---|---|---|---|
| 7 | $[x,y,1]$ | $[x,y,y^2,1]$ | 125 | 135 | 10 |
| 8 | $[x,y,1]$ | $[x,y,y^2,1]$ | 124 | 130 | 6 |
| 9 | $[x,y,1]$ | $[x,y,y^2,1]$ | 125 | 140 | 15 |
| 10 | $[x,y,1]$ | $[x,y,y^2,1]$ | 134 | 135 | 1 |
| 11 | $[x,y,1]$ | $[x,y,y^2,1]$ | 115 | 135 | 20 |
| 15 | $[x,y,1]$ | $[x,y,y^2,1]$ | 292 | 130 | −162 |
| 19 | $[x,y,1]$ | $[x,y,y^2,1]$ | 350 | 146 | −206 |
| 20 | $[x,y,1]$ | $[x,y,x^2,y^2,1]$ | 600 | 270 | −330 |
| 24 | $[x,y,z,1]$ | $[x,y,z,y^2,z^2,1]$ | 188 | 255 | 67 |
| 25 | $[x,y,z,1]$ | $[x,y,z,y^2,z^2,1]$ | 146 | 202 | 56 |
| 26 | $[x,y,z,n,1]$ | $[x,y,z,n,y^2,z^2,n^3,1]$ | 276 | 760 | 484 |

因为本文方法与文献[19]中的方法在最后一步都需要对候选秩函数进行验证，所以表 2 列出了两种方法对候选秩函数的验证时间。从表 2 中的数据可以看出，对大多数例子来说，因为本文方法能够得到线性模板秩函数，从而降低了验证阶段的计算复杂度，所以其在验证候选秩函数方面的时间代价比文献[19]的方法更低，并且随着变量的增加以及秩函数模板次数的增加，两者的时间差距也相应增大。

本文方法将区域划分与 SVM 方法相结合，可以使用形式

更简单的模板寻找秩函数，同时提高找到秩函数的可能性。

**结束语** 在循环程序的终止性判定问题上，目前研究较多的是寻找秩函数的方法。而该方法的关键在于对循环空间的处理和秩函数模板的选择。我们聚焦于第一个问题，根据循环空间中状态点的迭代轨迹，发现了一种新的划分方式。在判断循环终止性时，先判断该循环是否存在不变集，接着再寻找秩函数，使得程序的判定逻辑更加完善。

目前我们只探索到一类称为双向迭代的循环，针对这类循环，只需要将其程序空间划分为3个部分，但还可能存在三向、四向迭代等多向迭代的情况。在多向迭代中，可能会存在多个超平面，给划分带来困难。在这种情况下，获得的子区域不确定，从而导致秩函数在寻找上也存在着不确定性。我们将在后续的工作中对这一问题进行研究和解决。

本文利用机器学习的算法，将寻找秩函数问题转化为二分类问题。机器学习这一领域中还存在着其他算法，如线性回归，决策树等。在接下来的工作中，我们将继续应用其他机器学习算法，去寻找循环程序的秩函数。

## 参 考 文 献


[1] COL'ON M A, SIPMA H B. Synthesis of Linear Ranking Functions[C]// International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2001: 67-81.

[2] COL'ON M A, SIPMA H B. Practical Methods for Proving Program Termination[C]// International Conference on Computer Aided Verification. Berlin: Springer, 2002: 442-454.

[3] PODELSKI A, RYBALCHENKO A. A Complete Method for the Synthesis of Linear Ranking Functions[C]// International Workshop on Verification, Model Checking, and Abstract Interpretation. Berlin: Springer, 2004: 239-251.

[4] BRADLEY A R, MANNA Z, SIPMA H B. The Polyranking Principle[C]// International Colloquium on Automata, Languages, and Programming. Berlin: Springer, 2005: 1349-1361.

[5] BRADLEY A R, MANNA Z, SIPMA H B. Linear Ranking with Reachability[C]// Computer Aided Verification, 17th International Conference, CAV 2005. Edinburgh, Scotland: DBLP, 2005: 491-504.

[6] BEN-AMRAM A M, GENAIM S. Ranking Functions for Linear-Constraint Loops[J]. Journal of the ACM, 2012, 61(4): 1-55.

[7] BAGNARA R, ESNARD F, PESCETTI A, et al. A new look at the automatic synthesis of linear ranking functions[J]. Information and Computation, 2012, 215: 47-67.

[8] BEN-AMRAM A M, GENAIM S, MASUD A N. On the Termination of Integer Loops[J]. ACM Transactions on Programming Languages & Systems, 2012, 34(4): 1-24.

[9] BAGNARA R, MESNARD F. Eventual Linear Ranking Functions [C]// Proceedings of the 15th Symposium on Principles and Practice of Declarative Programming. Madrid, Spain: ACM, 2013: 229-238.

[10] BEN-AMRAM A M, GENAIM S. On the Linear Ranking Problem for Integer Linear-ConstraintLoops[J]. ACM Sigplan Notices, 2013, 48(1): 51-62.

[11] LEIKE J, HEIZMANN M, HOENICKE J, et al. Linear Ranking for Linear Lasso Programs. [C]// Automated Technology for Verification and Analysis. Cham: Springer, 2013: 365-380.

[12] LEIKE J, HEIZMANN M. Ranking Templates for Linear Loops [J]. Logical Methods in Computer Science, 2015, 11(1): 1-27.

[13] BEN-AMRAM A M, GENAIM S. On Multiphase-Linear Ranking Functions[C]// International Conference on Computer Aided Verification. Cham: Springer, 2017: 601-620.

[14] BEN-AMRAM A M, DOMÉNECH J J, GENAIM S. Multiphase-Linear Ranking Functions and their Relation to Recurrent Sets[C]// International Static Analysis Symposium. Cham: Springer, 2018: 459-480.

[15] COUSOT P. Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming[C]// Verification, Model Checking, & Abstract Interpretation. Berlin: Springer, 2005: 1-24.

[16] CHEN Y, XIA B, LU Y, et al. Discovering Non-linear Ranking Functions by Solving Semi-algebraic Systems[C]// Theoretical Aspects of Computing—ICTAC 2007. 2007: 34-49.

[17] SHEN L, WU M, YANG Z, et al. Generating exact nonlinear ranking functions by symbolic-numeric hybrid method[J]. Journal of Systems Science and Complexity, 2013, 26(2): 291-301.

[18] LI Y, ZHU G, FENG Y. The L-Depth Eventual Linear Ranking Functions for Single-Path Linear Constraint Loops[C]// The 10th International Symposium on Theoretical Aspects of Software Engineering. IEEE, 2016: 30-37.

[19] YUAN Y, LI Y. Ranking function detection via SVM: A more general method[J]. IEEE Access, 2019(7): 9971-9979.

[20] LI Y, SUN X C, LI Y, et al. Synthesizing Nested Ranking Functions for Loop Programs via SVM[C]// International Conference on Formal Engineering Methods. Cham: Springer, 2019: 438-454.

[21] SUN X C. Proving the Termination of Loop Programs based on SVM [D]. Beijing: Institute of Software, Chinese Academy of Sciences, 2020.

[22] GENAIM S. LoopKiller: website for termination analysis[EB/OL]. http://loopkiller.com/irankfinder.


**WANG Yao**, born in 1997, postgraduate. Her main research interest is program verification.

**LI Yi**, born in 1980, Ph. D, associate professor. His main research interests include program verification and symbolic computation.

（责任编辑：何杨）