学术写作与学术规范

学术论文写作

苏小红

哈尔滨工业大学 计算机学院

学术写作与学术论文的特点

学术写作与其他写作有何不同?

是个技术活

有套路

有规矩

有约束

有章法

学术写作与其他写作有何不同?

强套路性

- 这里说的套路
  - 不是指具体的模板
  - 而是约定俗成的文章结构、行文方法、逻辑脉络和整体框架

学术写作与其他写作有何不同?

冷静文风

- 客观
  - fact based （以事实为依托）
  - 而非opinion based （以观点为依托）
    - 不说“我认为”、“根据我的经验”、“我相信”。。。

学术写作与其他写作有何不同?

强功能性

明晰	强逻辑性	整体的合一性
把一件事情说明白的能力，正中靶心的能力	先说什么、后说什么，像剥洋葱和搭积木一样把复杂的事情掰开了、揉碎了展现给读者	前后照应，数据和理论的统一 结果对提出的问题给出了回答 文章贡献弥补了文献综述中讨论到的问题

学术写作与其他写作有何不同?

严谨性

- 一丝不苟，严丝合缝
- 体现在:
  - 小题大做
  - 逻辑缜密
  - 用词谨慎
    - 不说 Previous research **never** ...
    - 而说 Previous research **rarely** ...
  - 用词专业

学术论文的特点

- 创新性
- 学术性
- 科学性和准确性
- 规范性和可读性

论文的创新性

《Nature》

创新是指科研成果新颖，引人注意（出人意料或令人吃惊），而且该研究看来在该领域之外具有广泛的意义

《Science》

创新是指提出新见解，而不是对已有研究结论的再次论证，内容激动人心并富有启发性，具有广泛的科学兴趣

《中国科学》、《科学通报》、《自然科学进展》

在基础研究和应用研究方面具有创造性、高水平和有重要意义的最新研究成果

与《Nature》和《Science》的不同

未强调论文内容应能引起科学界广泛的兴趣
研究一定要深入，结果一定要深刻，要能反映研究者独到见解

论文的创新性

新领域、新方向、新理论、新方法、新应用

影响力递减

new

1

开辟一个全新的学科领域

一流学者
提出问题

new

2

提出一种全新的研究方法

二流学者
解决问题

better

3

提出一种改进的解决方法

三流学者
修修补补
补充、完善、扩展

绝不做四流学者抄来抄去！

论文的学术性

教科书

介绍和传授已有知识，

主要读者是外行人、初学者，

强调系统性、完整性和连续性，

写法上要力求循序渐近，深入浅出，由浅入深

工作总结

详细介绍做了哪些工作，怎么做的，实验过程和操作以及数据资料，有什么优缺点，有什么经验和体会

重复别人的工作也可以写进去

论文的学术性

学术论文

与他人相重复的研究内容、基础性知识、 一般性实验过程或数学推导、比较浅显的分析等都应删去, 或只作简要的交待和说明

应对原始材料有整理、有取舍、有提高

要形成新观点、新认识、新结论，有学术价值

对实验或用其他方式得到的结果, 从一定的理论高度分析和总结, 形成一定的科学见解, 包括提出并解决一些有科学价值的问题

对自己提出的科学见解或问题, 要用事实和理论进行符合逻辑的论证、分析或说明, 总之要将实践上升为理论

论文的科学性和正确性

科学性

正确说明研究对象所具有的特殊矛盾, 要尊重事实, 尊重科学

包括：论点正确、论据必要而充分、论证严密、推理符合逻辑、数据可靠、计算准确、结果深刻、有启发性 、结论可靠等

正确性

对客观事物，即研究对象的规律和性质，表述的接近程度。

包括：概念、定义、判断、分析和结论要准确，对自己研究成果的评估要确切、恰当，对他人研究成果的对比评价要实事求是, 切忌片面性和说过头话

论文的规范性和可读性

可读性

文字表达：

语言简明，文字通顺，行文流畅，清楚易懂，无语法错误

结构严谨，层次分明，论述完整，逻辑严密，条理清晰

让其他领域读者和母语为非英语的读者也能读懂

规范性

技术表达：

专业术语准确，前后一致

图表、公式、体例、格式规范，文字和插图配合恰当

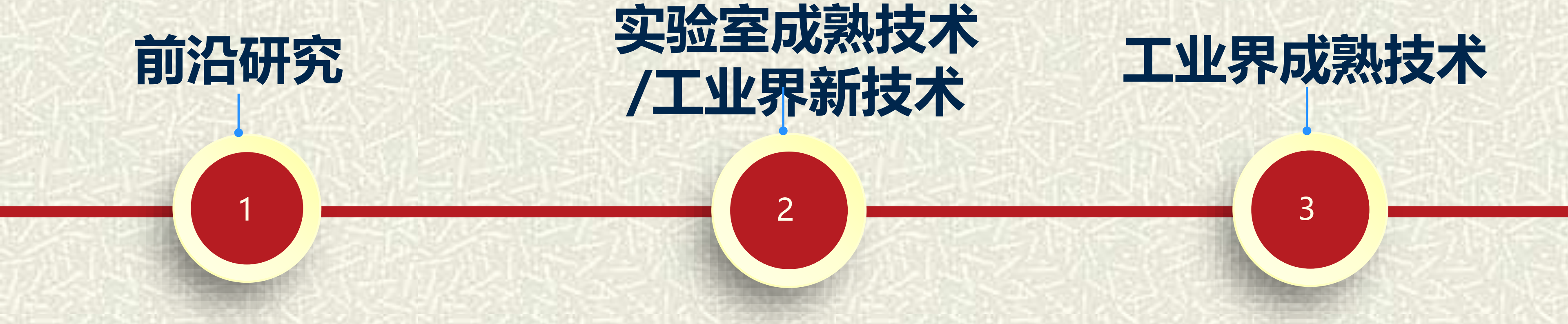
看看 Nature 对可读性的要求

- Nature在投稿指南对可读性的要求：
  - 来稿应写得清楚、简练
    - 以便让其他领域的读者和母语为非英语的读者能读懂
  - 基本的专业术语应作简明解释，但不是说教式的
- Nature杂志一再申明：
  - 论文是否利用最终决定权属于编辑部，凡不符合可读性要求的稿件，毋需送审即行退稿

为什么要写学术论文?

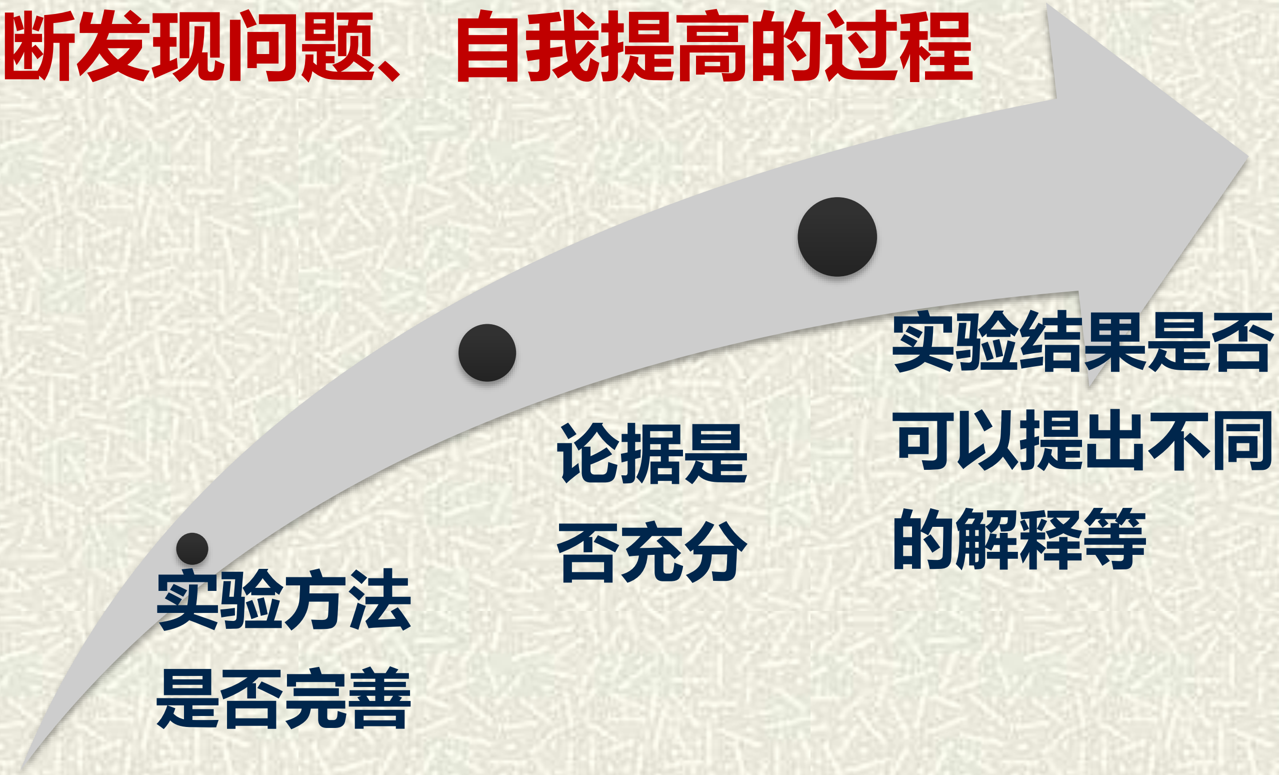
为什么要写学术论文?

- 把你的工作告诉同行
- 成果转化为应用



为什么要写学术论文?

- 对自己工作的全面总结和深入思考
- 不断发现问题、自我提高的过程



实验方法是否完善

论据是否充分

实验结果是否可以提出不同的解释等

发表学术论文那些事

- 让人又爱又恨的事
- 从提交到发表，需经历

同行评审 → 编辑对文字和格式的苛求 → 一改再改的漫长过程 → 最终得以发表

any

How to write a great research paper ✕ Do not wait! Write

综述性学术论文

综述性学术论文的结构

1. 选定研究领域
2. 广泛查阅文献
3. 整理研究脉络
4. 归纳国内外研究现状
5. 分类、对比
6. 展望

1. 引言
2. 问题与挑战
3. 技术方向分类
4. 研究现状分析
5. 未来研究展望
6. 结论

综述性学术论文撰写的整体要求

- （1）综述应有创新性
- （2）选题应反映当前最新研究热点
- （3）避免与已有综述论文重复，突出本文侧重点
- （4）尽量贴近本人研究方向，有相关工作基础
- （5）引用文献要全面
- （6）文字结合图表
- （7）主题明确、鲜明，内容不宜过于宽泛

综述性学术论文撰写的具体要求

- （1）应有合理的相关工作分类框架
- （2）形成有条理的领域知识体系，便于同行理解掌握
- （3）根据论述需要，明确给出核心概念
- （4）明确给出当前领域的问题与挑战
- （5）研究现状分析比较和评述深入、深刻
- （6）在分析基础上，给出未来研究方向
- （7）应有对应用需求、趋势及现状等问题的分析
- （8）建议重点对近5年的最新工作进行分析

学术论文的基本结构

学术论文的基本构成

- **前置部分**
  - 题目
  - 作者
  - 摘要
  - 关键词
- **主体部分**
  - 引言
  - 正文
  - 结论
  - 参考文献

英文学术论文的基本结构

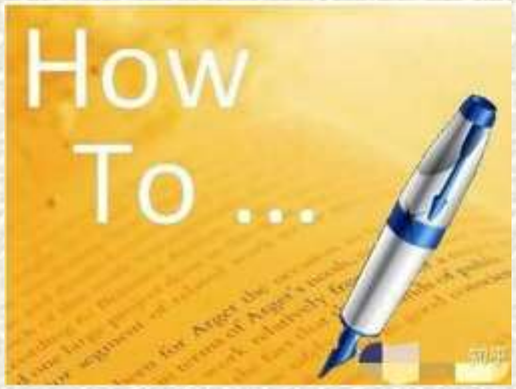
- Outline
  - Title
  - (1) Abstract
  - (2) Introduction（含Your contribution）
  - (3) Previous Work（Related Work）
  - (4) Our Solution（Motivation and theoretical Support）
  - (5) Experiments （Experimental Support）
  - (6) Discussion（Performance Analysis，Threats To Validity）
  - (7) Conclusion
  - (8) Acknowledgement（Optional）
  - (9) References
  - (10) Appendix（Optional）

正文结构不是千篇一律

写作形式可作适当调整

撰写学术论文的要点

- 你需要说的其实就是这些：
  - Problem X is important
  - Previous works A, B, and C have been done
  - A, B, and C have their weakness
  - Your work D
  - Theoretic analysis
  - Experimental comparison against A, B, and C
  - Why D is better
  - Strength and weakness of D
  - Future works on D



标题（Title）

论文标题的重要性

- Title
  - 应能清晰地描述本文的研究内容，又能体现与其他文献的区别

论文标题的基本要求

- **准确得体**
  - 准确表达论文的中心内容和创新性
  - 要善于给论文的创新点命名，并体现在论文的题目中
- **简短精练**
  - 某些期刊对标题中包含的字符数有所限制
    - GB7713-87 规定题目“一般不宜超过 20 字”
    - 国外出版的英文科技期刊一般要求论文题目不超过 12 个词
- **便于检索**
  - 大多数搜索引擎会使用关键词来定位标题相关的论文
- **容易认读**
  - 题名中应当避免使用非共知共用的缩略词、字符、代号等

论文标题的较高要求

记忆深刻，过目不忘

特点鲜明

有吸引力

如何使题目便于检索?

- 题目所用词语，必须有助于选定关键词和为检索提供特定的实用信息
- 题目中一定要有反映文章内容的关键词
  - SCI 网络版检索中有一个选项就是“限定在题目中检索关键词”
  - 一般不用学科或分支学科的科目作为题目组成部分

论文标题的语法要求

详略得当

用词准确

结构合理

论文标题的**语法**要求

- **结构合理**
  - 习惯上题名**不用动宾结构**，而**用以名词或名词性词组为中心的偏正短语**（定、状语等修饰语+中心语）
  - **定语词组的类型选择不当**，有时会产生歧义
    - e.g.研究xx行为的几个基本理论问题
    - ？研究xx行为的问题
    - ？研究xx行为时用到的几个理论问题
    - →xx行为研究中的几个基本理论问题

论文标题的**语法**要求

- **结构合理**
  - 习惯上题名**不用动宾结构**，而**用以名词或名词性词组为中心的偏正短语**（定、状语等修饰语+中心语）
  - **语序不对**，有时造成语意混乱，使人不知所云
    - 深度神经网络稀疏性的软硬件联合优化方法研究
    - →软硬件联合优化的深度神经网络稀疏性研究
    - 影响xx的因素分析
    - →xx的影响因素分析

论文标题的语法要求

- 详略得当
  - 不随便省略词语
  - 避免“的”的多用或漏用
    - 联合词组、偏正词组、主谓词组、动宾词组、介词词组做定语时, 中心语之前需用“的”
    - 而修辞规则又要求：多项定语中的“的”字不宜多用

软硬件联合优化深度神经网络稀疏性研究

软硬件联合优化**的**深度神经网络**的**稀疏性研究

软硬件联合优化深度神经网络**的**稀疏性研究

软硬件联合优化**的**深度神经网络稀疏性研究

论文标题的常见问题

- 题目涉及的面过大过宽
  - e.g. 对话系统关键技术研究
  - e.g.行人目标检测算法研究
- 题目过于一般化，无特点
  - e.g. 改进的行人目标检测算法
  - →基于多摄像头的行人视频运动目标检测算法
  - →基于三维模型的行人徘徊行为检测算法
  - →一种改进的深度残差网络行人检测方法
- 语法、修辞、逻辑上，有语病

短标题有什么好处?

- 有可能获得更高的被引次数
  - 论文的标题越短，被引次数越高
  - 相对于新发表的论文，“标题越短，引用越多”的现象在老论文中更常见

Science标题最长的两篇论文，2010年至今的引用量分别是68和67

“The role ofparticle morphology in interfacial energy transfer in CDSE/CDS heterostructurenanocrystals ”

“Insects betray themselves in nature to predators byrapid isomerization of green leaf volatiles”

Science标题最短的两篇论文，引用量分别是253和700

“Quantum walks of correlated photos”

“A draft sequence of the neandertal genome”

短标题论文为什么会高被引?

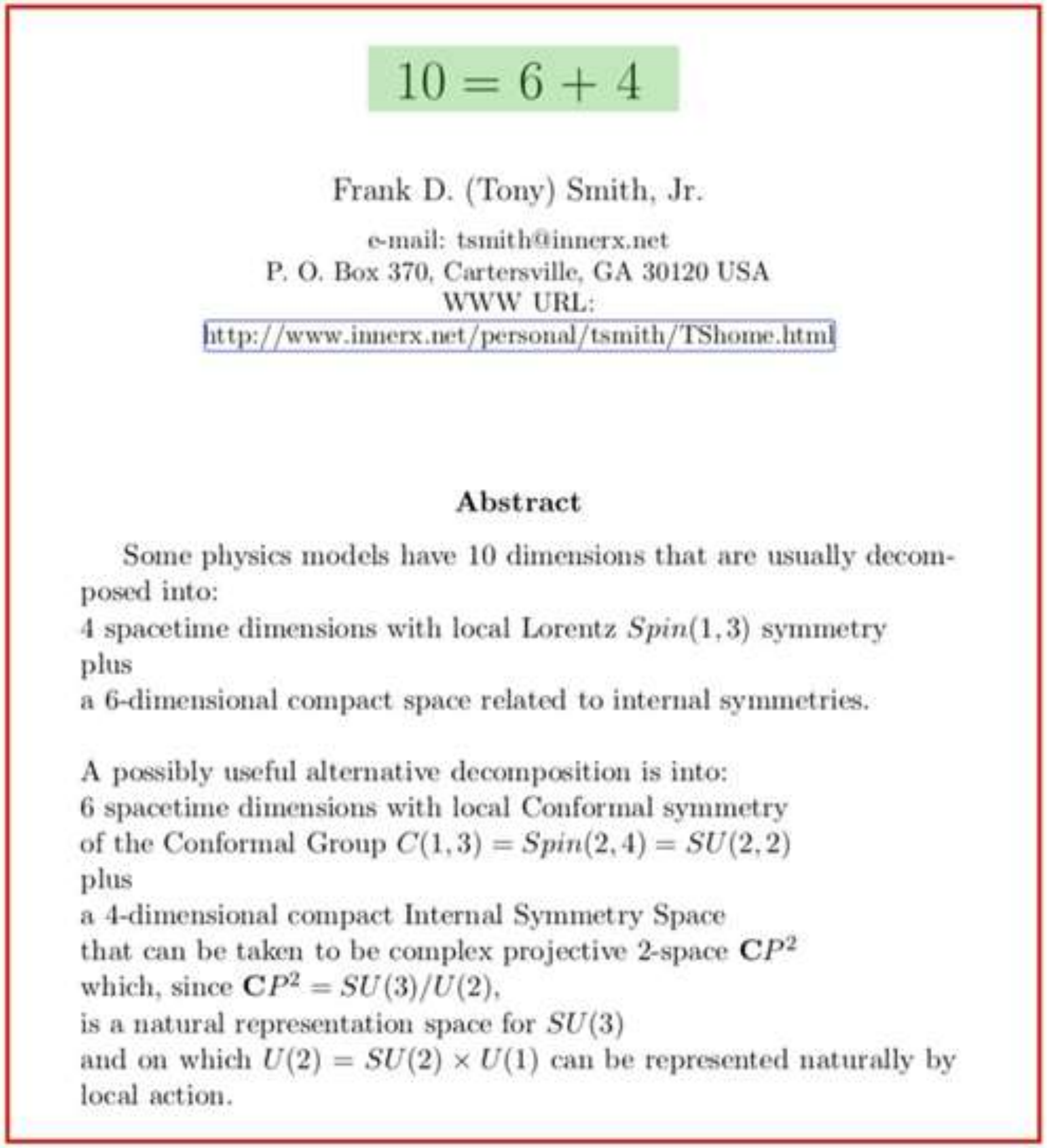
- 标题短的论文，内容更易理解，更吸引眼球
- 期刊因素
  - 高水平期刊对论文的要求更严格，甚至会限制论文标题的长度
  - 意味着，有一部分高水平论文的作者，可能是被迫选择了短标题
- 并非适用于所有学科
  - 特例：在医学类刊物中，长标题论文获得了更多被引次数
- 莫为追求短而导致大而空

短标题是如何练成的?

- 短标题通常都很牛
- An example of title
  - “甲地区乙时期丙昆虫交配过程的跟踪研究及其结果分析”
  - “昆虫交配后吃掉配偶的原因探讨”
  - “昆虫求偶过程中的献身行为”
  - “Eat me or Die”

The title should be both brief and descriptive. Introductory words, such as "On,'' "Observations on," "Some," and "Study of'' are to be avoided

最牛短标题示例



$10 = 6 + 4$

Frank D. (Tony) Smith, Jr.

e-mail: tsmith@innerx.net
P. O. Box 370, Cartersville, GA 30120 USA
WWW URL:
http://www.innerx.net/personal/tsmith/TShome.html

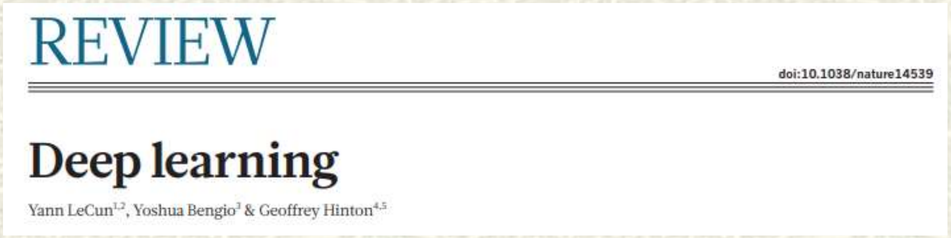
Abstract

Some physics models have 10 dimensions that are usually decomposed into:
4 spacetime dimensions with local Lorentz $Spin(1,3)$ symmetry
plus
a 6-dimensional compact space related to internal symmetries.

A possibly useful alternative decomposition is into:
6 spacetime dimensions with local Conformal symmetry
of the Conformal Group $C(1,3) = Spin(2,4) = SU(2,2)$
plus
a 4-dimensional compact Internal Symmetry Space
that can be taken to be complex projective 2-space $\mathbf{C}P^2$
which, since $\mathbf{C}P^2 = SU(3)/U(2)$,
is a natural representation space for $SU(3)$
and on which $U(2) = SU(2) \times U(1)$ can be represented naturally by local action.

CS领域短标题示例

- World Model（2018)
- Fisher GAN
- Bayesian GAN
- Deep Signatures
- Attention is All you Need
- Online multiclass boosting
- History Driven Program Repair
- Noise-tolerant fair classification
- Multi-View Reinforcement Learning
- Systems that listen, speak and understand
- Deep contextualized word representations
- Delayed Impact of Fair Machine Learning
- Mining StackOverflow for Program Repair
- Deep Learning without Weight Transport
- Is Deeper Better only when Shallow is Good?
- GenProg: A Generic Method for Automatic Software Repair
- DeepFix: Fixing Common C Language Errors by Deep Learning
- Deep Residual Learning for Image Recognition (2016)
- Adversarial Examples Are Not Bugs, They Are Features（2019)



REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

摘要（Abstract）

摘要写什么?

- 标题和摘要
  - 目的都是提高文章的引用率
- Abstract
  - 对自己工作及其贡献的总结
    - a)阐述问题
    - b)问题的解决方案和结果

中文摘要的四要素

- **目的：要解决的问题，问题的重要性，目前存在的不足解决问题的动机**
  针对….存在的**问题**
- **方法：采用的理论、平台、工具、过程、手段等**
  基于的….理论，提出了…方法，并在…平台上，采用…工具，实现了…系统
- **结果：研究的成果/实验的数据，得到的性能、效果**
  经过…数据集的测试，本文提出的…方法，相比…方法，在…方面提高了…
- **结论：结果的分析、比较、评价、应用效果、影响和意义**
  本文的…方法具有…特点，在…方面有显著的提高，具有重要的推广应用价值

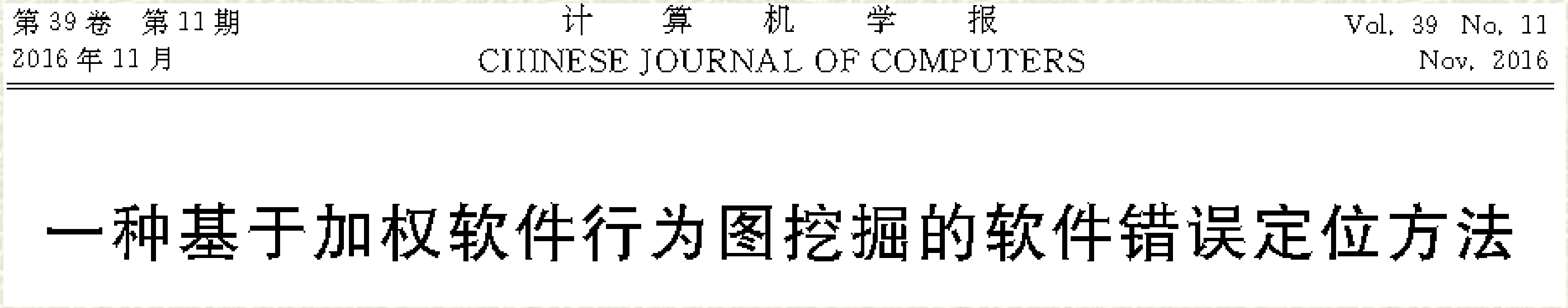
摘要的撰写要求

- 四要素的权重——**哪个重要，就突出哪个**
  - 如果某个要素完全不重要，那么可以不出现
- 方法
  - 论文采用的手段、过程和方法，必要时介绍使用手段的方式
  - 尽量指出可靠性和适应的参数范围
- 结果
  - 尽可能给出量化结果
    - “不一样”不如“高、低、强、弱”
    - “高、低、强、弱”不如“高多少、低多少、强多少、弱多少”
- 结论
  - 除交代解决了什么问题外，还要总结对本领域或其他领域的通用价值

中文摘要示例

- 摘要：（目的）为改善钢铁材料的耐高温腐蚀性能, 用光束合金化方法在45钢表面合成了Fe-Al金属间化合物涂层. （过程和方法）采用扫描电子显微镜、能量弥散X射线分析和X射线衍射研究了光束合金化工艺参数(粉末预置量m和热输入量q)对合金化层的化学成分、显微组织及其物相组成的影响. （结果）实验结果表明: 减小比能量(E = q/m)将导致合金化层的熔宽和熔深减小, 从而使合金化层含Fe量减少, 含Al量增加;该实验条件下, 获得了Fe与Al原子数比为2.4～19.2的合金化层. 由比能量决定的Fe与Al原子数比是合金化层显微组织及其物相组成的重要影响因素, 合金化层的显微组织有α-Fe固溶体、α-Fe固溶体+Fe3Al金属间化合物及FeAl+AlFe3C0.5金属间化合物3种类型. （结论）降低热输入或增加粉末预置量均可引起合金化层中Fe与Al原子数比的降低,有助于Fe-Al系(Fe3Al或FeAl)金属间化合物的合成.

研究型论文的摘要示例

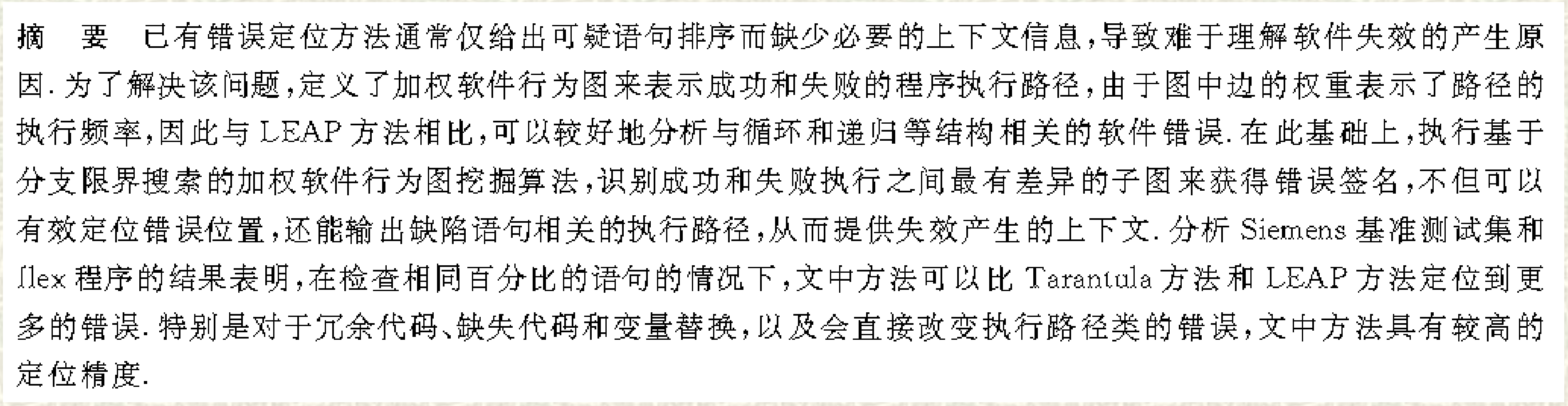


第 39 卷　第 11 期
2016 年 11 月

计　　算　　机　　学　　报
CHINESE JOURNAL OF COMPUTERS

Vol. 39　No. 11
Nov. 2016

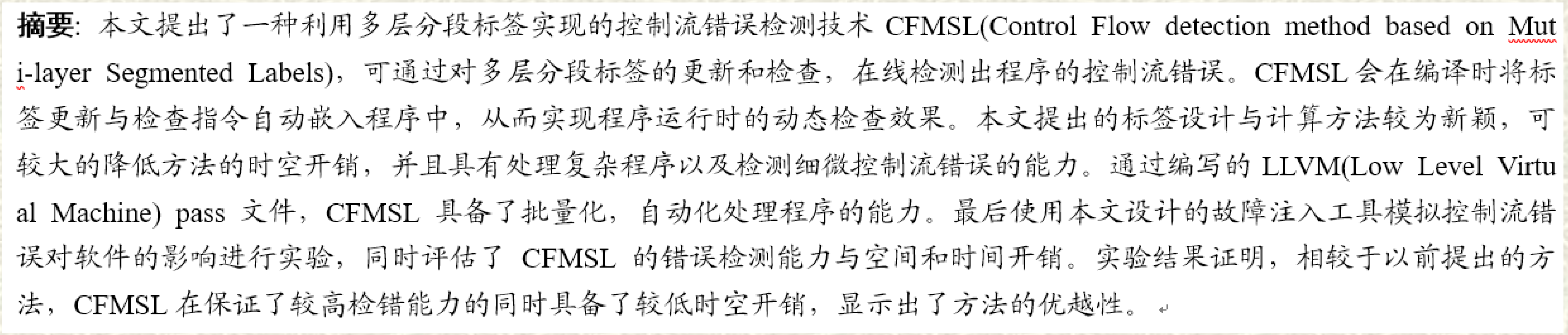
一种基于加权软件行为图挖掘的软件错误定位方法



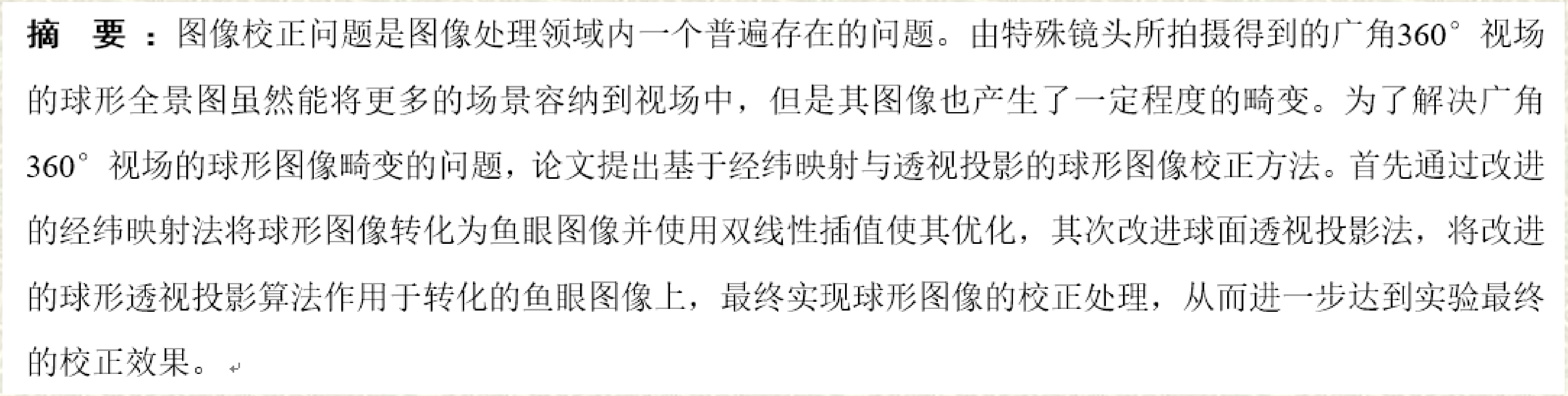
摘　要　已有错误定位方法通常仅给出可疑语句排序而缺少必要的上下文信息，导致难于理解软件失效的产生原因．为了解决该问题，定义了加权软件行为图来表示成功和失败的程序执行路径，由于图中边的权重表示了路径的执行频率，因此与 LEAP 方法相比，可以较好地分析与循环和递归等结构相关的软件错误．在此基础上，执行基于分支限界搜索的加权软件行为图挖掘算法，识别成功和失败执行之间最有差异的子图来获得错误签名，不但可以有效定位错误位置，还能输出缺陷语句相关的执行路径，从而提供失效产生的上下文．分析 Siemens 基准测试集和 flex 程序的结果表明，在检查相同百分比的语句的情况下，文中方法可以比 Tarantula 方法和 LEAP 方法定位到更多的错误．特别是对于冗余代码、缺失代码和变量替换，以及会直接改变执行路径类的错误，文中方法具有较高的定位精度．

请找下4要素：目的、过程和方法、结果、结论

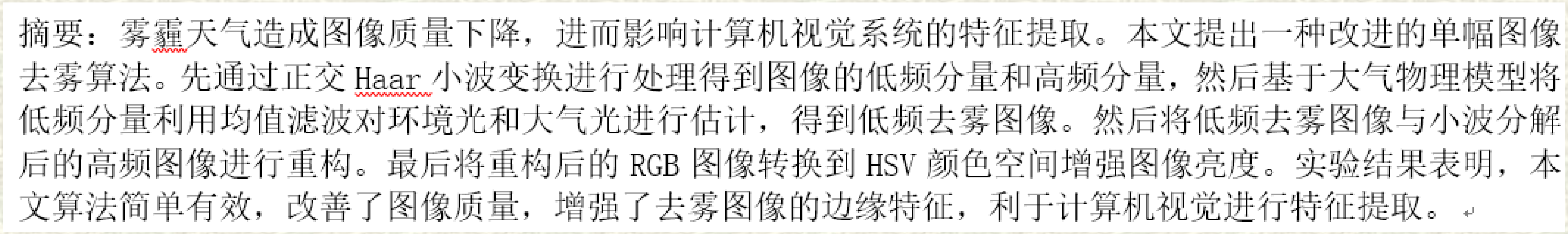
中文摘要问题示例



摘要: 本文提出了一种利用多层分段标签实现的控制流错误检测技术 CFMSL(Control Flow detection method based on Muti-layer Segmented Labels)，可通过对多层分段标签的更新和检查，在线检测出程序的控制流错误。CFMSL 会在编译时将标签更新与检查指令自动嵌入程序中，从而实现程序运行时的动态检查效果。本文提出的标签设计与计算方法较为新颖，可较大的降低方法的时空开销，并且具有处理复杂程序以及检测细微控制流错误的能力。通过编写的 LLVM(Low Level Virtual Machine) pass 文件，CFMSL 具备了批量化，自动化处理程序的能力。最后使用本文设计的故障注入工具模拟控制流错误对软件的影响进行实验，同时评估了 CFMSL 的错误检测能力与空间和时间开销。实验结果证明，相较于以前提出的方法，CFMSL 在保证了较高检错能力的同时具备了较低时空开销，显示出了方法的优越性。↵



摘　要：图像校正问题是图像处理领域内一个普遍存在的问题。由特殊镜头所拍摄得到的广角360°视场的球形全景图虽然能将更多的场景容纳到视场中，但是其图像也产生了一定程度的畸变。为了解决广角360°视场的球形图像畸变的问题，论文提出基于经纬映射与透视投影的球形图像校正方法。首先通过改进的经纬映射法将球形图像转化为鱼眼图像并使用双线性插值使其优化，其次改进球面透视投影法，将改进的球形透视投影算法作用于转化的鱼眼图像上，最终实现球形图像的校正处理，从而进一步达到实验最终的校正效果。↵



摘要：雾霾天气造成图像质量下降，进而影响计算机视觉系统的特征提取。本文提出一种改进的单幅图像去雾算法。先通过正交 Haar 小波变换进行处理得到图像的低频分量和高频分量，然后基于大气物理模型将低频分量利用均值滤波对环境光和大气光进行估计，得到低频去雾图像。然后将低频去雾图像与小波分解后的高频图像进行重构。最后将重构后的 RGB 图像转换到 HSV 颜色空间增强图像亮度。实验结果表明，本文算法简单有效，改善了图像质量，增强了去雾图像的边缘特征，利于计算机视觉进行特征提取。↵

英文摘要的五要素

- Context/Background:
- Objective/Aim:
  - To …(不定式语句结构)
- Methods:
  - (被动语态，过去时态)
- Results:
  - (被动语态，过去时态)
- Conclusion:
  - (主动语态，现在时态)

- Purpose
  - Summarize your contributions
- Style
  - What is the problem
  - What is your work
  - Features of your work
  - Advantages of your work
  - Results

英文摘要的基本要求

- 英文摘要的文字要求：
  - 1） 第一句应避免与题目（Title）重复
  - 2） 尽量使用简单句
  - 3） 尽量使动词靠近主语
  - 4） 尽量不用第一人称作主语，也不用第三人称
  - 5） 以重要的事实开头，而不以辅助从句开头
  - 6） 在有动作主体的情况下，使用主动语态
  - 7）缩略语应加以说明，缩略语应先写全称，再写简称

中英文摘要撰写的通用要求

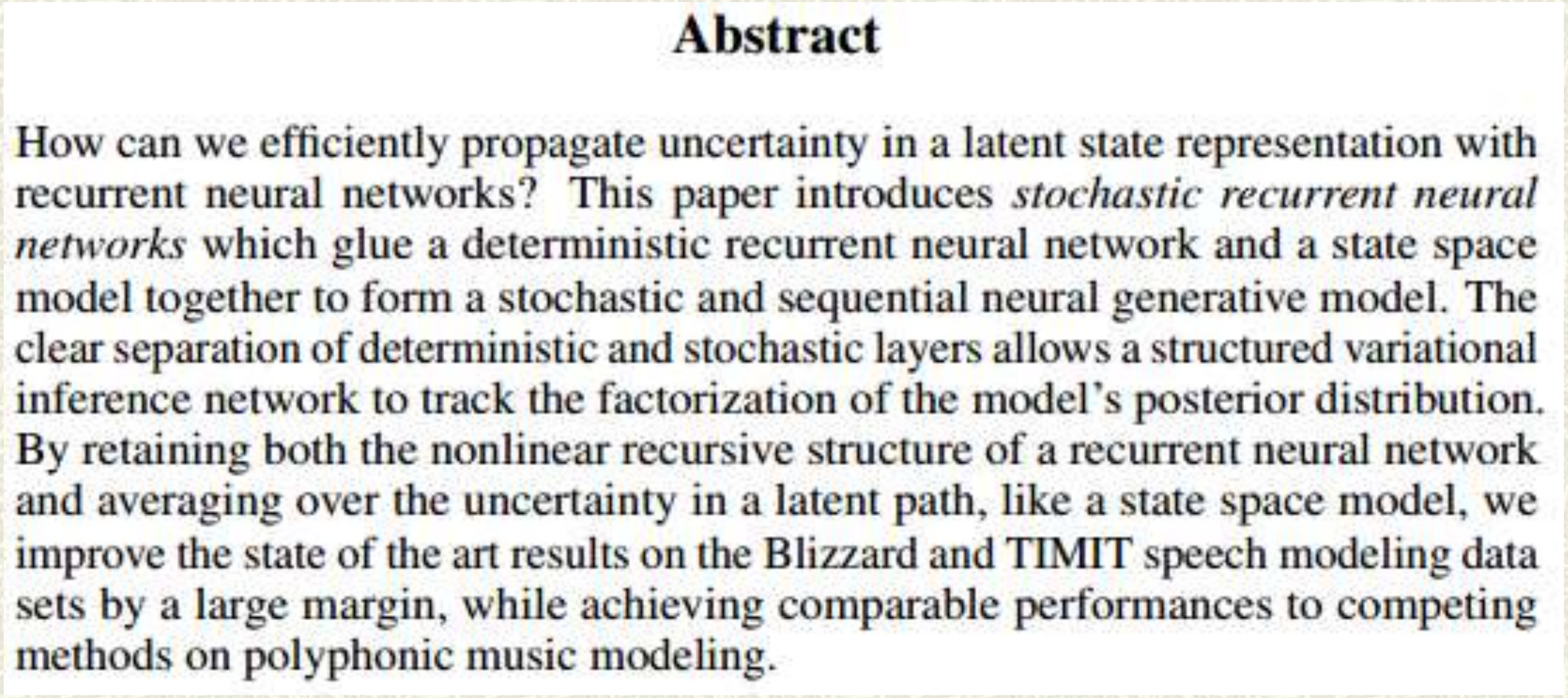
- Why、How、What
- 格式规范
  - 不要分段
  - 不要引用参考文献
  - 不要引用图表
  - 不要插入复杂的公式
- 语言通顺，简短精炼
  - 尽量避免带感情色彩的形容词
  - 避免言过其实、空洞无物
- 每次修订或变更手稿时，都应相应地修改摘要

Ei数据库对文摘的要求

- 长度一般150-200 words
- 取消不必要的字句，不说没有信息量的无用的话
  - It is reported …
  - Extensive investigations show that…
  - The author discusses …
  - This paper concerned with …
  - 文摘开头的 In this paper
  - “in detail”、 “briefly”、 “here”、 “new”、 “mainly”
  - “本文所谈的有关研究工作是对过去老工艺的一个极大的改进”，
  - “本工作首次实现了…”
  - “经检索尚未发现与本文类似的文献”

开门见山

- **尽量减少Background Information**
  - **Example paper: Sequential Neural Models with Stochastic layers（NIPS oral)**



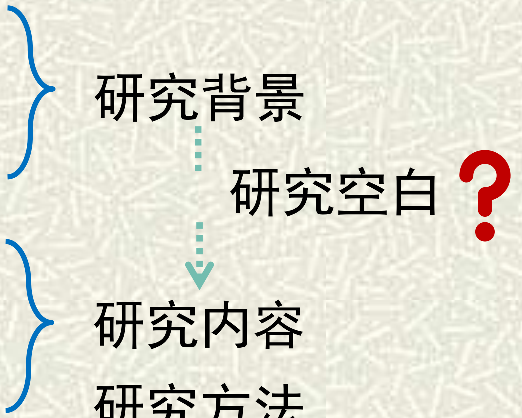
Abstract

How can we efficiently propagate uncertainty in a latent state representation with recurrent neural networks? This paper introduces *stochastic recurrent neural networks* which glue a deterministic recurrent neural network and a state space model together to form a stochastic and sequential neural generative model. The clear separation of deterministic and stochastic layers allows a structured variational inference network to track the factorization of the model's posterior distribution. By retaining both the nonlinear recursive structure of a recurrent neural network and averaging over the uncertainty in a latent path, like a state space model, we improve the state of the art results on the Blizzard and TIMIT speech modeling data sets by a large margin, while achieving comparable performances to competing methods on polyphonic music modeling.

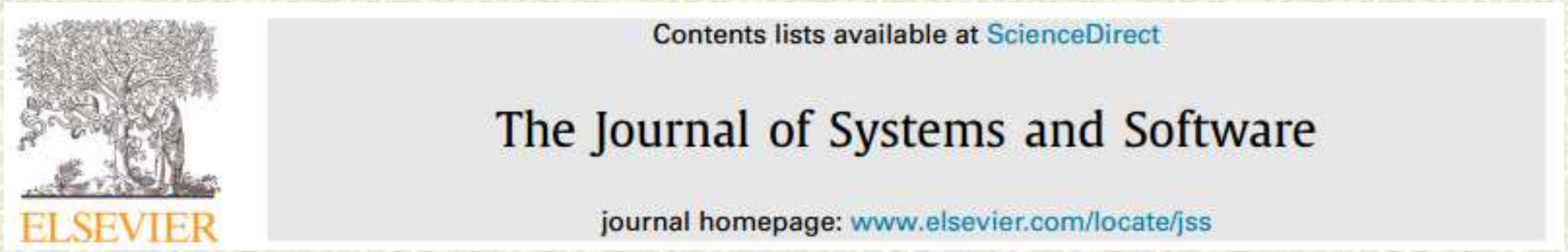
有问题的英文摘要示例

研究背景 → 研究空白 → 研究方法 研究内容 → 研究发现 → …

Low rank coal oxidation is the source of heat which may induce spontaneous combustion. Adsorbed water in low rank coal has been demonstrated favorable for this process. By combining density functional calculations and DSC experiments, this work provides a novel insight into the promotional roles of adsorbed water in low rank coal oxidation. Density functional calculations reveals that···



英文摘要示例

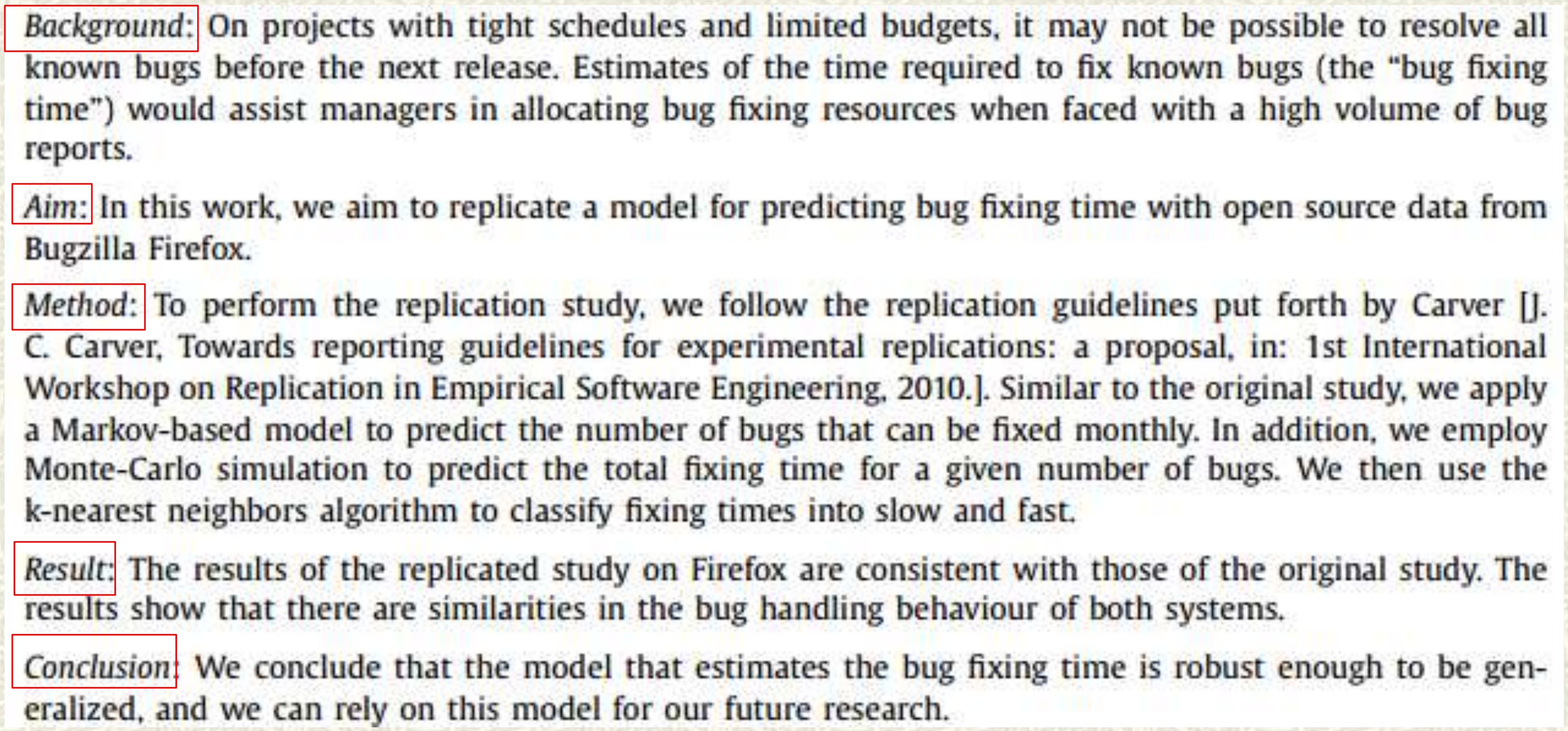


ELSEVIER

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss



Background: On projects with tight schedules and limited budgets, it may not be possible to resolve all known bugs before the next release. Estimates of the time required to fix known bugs (the "bug fixing time") would assist managers in allocating bug fixing resources when faced with a high volume of bug reports.

Aim: In this work, we aim to replicate a model for predicting bug fixing time with open source data from Bugzilla Firefox.

Method: To perform the replication study, we follow the replication guidelines put forth by Carver [J. C. Carver, Towards reporting guidelines for experimental replications: a proposal, in: 1st International Workshop on Replication in Empirical Software Engineering, 2010.]. Similar to the original study, we apply a Markov-based model to predict the number of bugs that can be fixed monthly. In addition, we employ Monte-Carlo simulation to predict the total fixing time for a given number of bugs. We then use the k-nearest neighbors algorithm to classify fixing times into slow and fast.

Result: The results of the replicated study on Firefox are consistent with those of the original study. The results show that there are similarities in the bug handling behaviour of both systems.

Conclusion: We conclude that the model that estimates the bug fixing time is robust enough to be generalized, and we can rely on this model for our future research.

英文摘要示例

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

ELSEVIER

Context: Identifying defects in code early is important. A wide range of static code metrics have been evaluated as potential defect indicators. Most of these metrics offer only high level insights and focus on particular pre-selected features of the code. None of the currently used metrics clearly performs best in defect prediction.

Objective: We use Abstract Syntax Tree (AST) n-grams to identify features of defective Java code that improve defect prediction performance.

Method: Our approach is bottom-up and does not rely on pre-selecting any specific features of code. We use non-parametric testing to determine relationships between AST n-grams and faults in both open source and commercial systems. We build defect prediction models using three machine learning techniques.

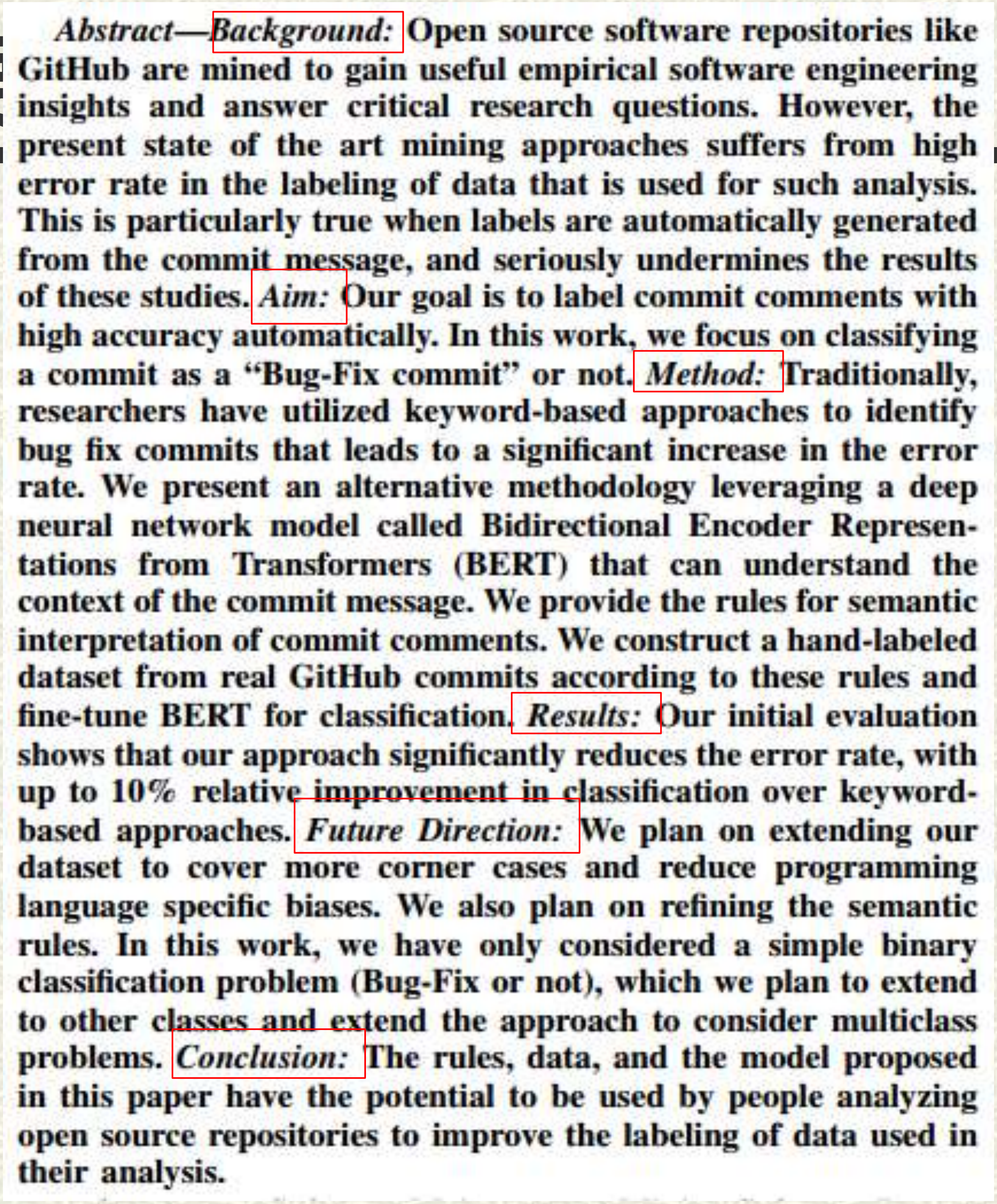
Results: We show that AST n-grams are very significantly related to faults in some systems, with very large effect sizes. The occurrence of some frequently occurring AST n-grams in a method can mean that the method is up to three times more likely to contain a fault. AST n-grams can have a large effect on the performance of defect prediction models.

Conclusions: We suggest that AST n-grams offer developers a promising approach to identifying potentially defective code.

英文摘要

⌗ 不同期刊有不同风格

⌗ 不同作者对各个要素的侧重点也有所不同



Abstract—*Background:* Open source software repositories like GitHub are mined to gain useful empirical software engineering insights and answer critical research questions. However, the present state of the art mining approaches suffers from high error rate in the labeling of data that is used for such analysis. This is particularly true when labels are automatically generated from the commit message, and seriously undermines the results of these studies. *Aim:* Our goal is to label commit comments with high accuracy automatically. In this work, we focus on classifying a commit as a "Bug-Fix commit" or not. *Method:* Traditionally, researchers have utilized keyword-based approaches to identify bug fix commits that leads to a significant increase in the error rate. We present an alternative methodology leveraging a deep neural network model called Bidirectional Encoder Representations from Transformers (BERT) that can understand the context of the commit message. We provide the rules for semantic interpretation of commit comments. We construct a hand-labeled dataset from real GitHub commits according to these rules and fine-tune BERT for classification. *Results:* Our initial evaluation shows that our approach significantly reduces the error rate, with up to 10% relative improvement in classification over keyword-based approaches. *Future Direction:* We plan on extending our dataset to cover more corner cases and reduce programming language specific biases. We also plan on refining the semantic rules. In this work, we have only considered a simple binary classification problem (Bug-Fix or not), which we plan to extend to other classes and extend the approach to consider multiclass problems. *Conclusion:* The rules, data, and the model proposed in this paper have the potential to be used by people analyzing open source repositories to improve the labeling of data used in their analysis.

英文摘要示例

- Abstract—Automatically detecting software vulnerabilities is an important problem that has attracted much attention. However, existing vulnerability detectors still cannot achieve the vulnerability *detection capability* and *locating precision* that would warrant their adoption for real-world use. In this paper, we present Vulnerability Deep learning-based Locator (VulDeeLocator), a deep learning-based fine-grained vulnerability detector, for C programs with source code. VulDeeLocator advances the state-of-the-art by simultaneously achieving a high detection capability and a high locating precision.

英文摘要示例（续）

When applied to three real-world software products, VulDeeLocator detects four vulnerabilities that are not reported in the National Vulnerability Database (NVD); among these four vulnerabilities, three are not known to exist in these products until now, but the other one has been "silently" patched by the vendor when releasing newer versions of the vulnerable product.

The core innovations underlying VulDeeLocator are (i) the leverage of intermediate code to accommodate semantic information that cannot be conveyed by source code-based representations, and (ii) the concept of *granularity refinement* for precisely pinning down locations of vulnerabilities

英文摘要示例

In recent years, automatic program repair has been a hot research topic in the software engineering community, and many approaches have been proposed. Although these approaches produce promising results, some researchers criticize that existing approaches are still limited in their repair capability, due to their limited repair templates. Indeed, it is quite difficult to design effective repair templates. An award-wining paper analyzes thousands of manual bug fixes, but summarizes only ten repair templates. Although more bugs are thus repaired, recent studies show such repair templates are still insufficient. We notice that programmers often refer to Stack Overflow, when they repair bugs. With years of accumulation, Stack Overflow has millions of posts that are potentially useful to repair many bugs. The observation motives our work towards mining repair templates from Stack Overflow.

重点在写背景和目的

英文摘要示例（续）

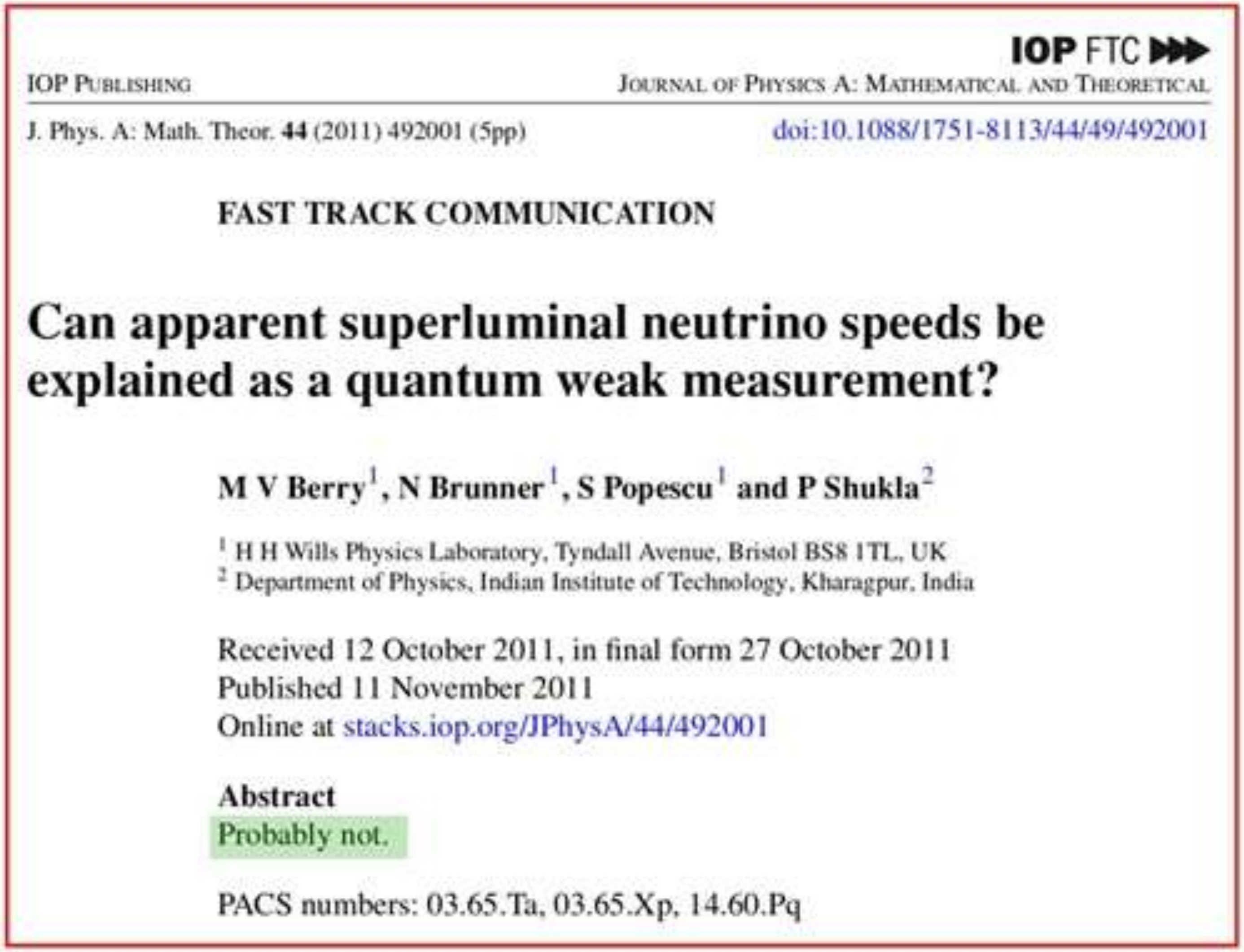
In this paper, we propose a novel approach, called SOFix, that extracts code samples from *Stack Overflow*, and *mines repair* patterns from extracted code samples. Based on our mined repair patterns, we derived 13 repair templates. We implemented these repair templates in SOFix, and conducted evaluations on the widely used benchmark, Defects4J.

Our results show that SOFix repaired 23 bugs, which are more than existing approaches. After comparing repaired bugs and templates,

we find that SOFix repaired more bugs, since it has more repair templates. In addition, our results also reveal the urgent need for better fault localization techniques.

Mining stackoverflow for program repair

最摘要


IOP PUBLISHING IOP FTC ▶▶▶
JOURNAL OF PHYSICS A: MATHEMATICAL AND THEORETICAL

J. Phys. A: Math. Theor. **44** (2011) 492001 (5pp) doi:10.1088/1751-8113/44/49/492001

FAST TRACK COMMUNICATION

Can apparent superluminal neutrino speeds be explained as a quantum weak measurement?

M V Berry[1], N Brunner[1], S Popescu[1] and P Shukla[2]

[1] H H Wills Physics Laboratory, Tyndall Avenue, Bristol BS8 1TL, UK
[2] Department of Physics, Indian Institute of Technology, Kharagpur, India

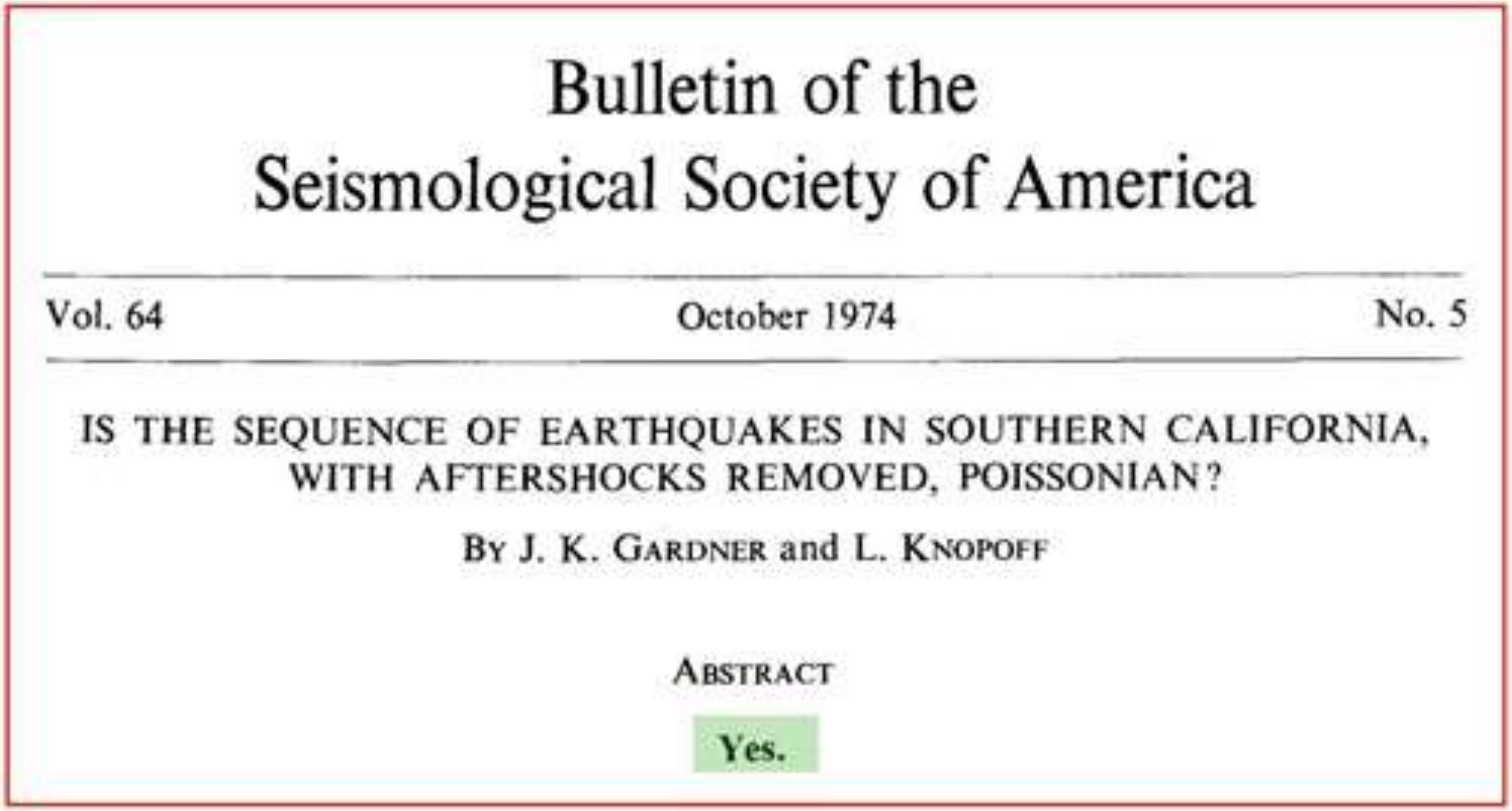
Received 12 October 2011, in final form 27 October 2011
Published 11 November 2011
Online at stacks.iop.org/JPhysA/44/492001

Abstract
Probably not.

PACS numbers: 03.65.Ta, 03.65.Xp, 14.60.Pq

中微子超光速现象可以用量子弱测量解释吗？
应该不可能（Probably not)

最摘要



Bulletin of the
Seismological Society of America

Vol. 64 October 1974 No. 5

IS THE SEQUENCE OF EARTHQUAKES IN SOUTHERN CALIFORNIA, WITH AFTERSHOCKS REMOVED, POISSONIAN?

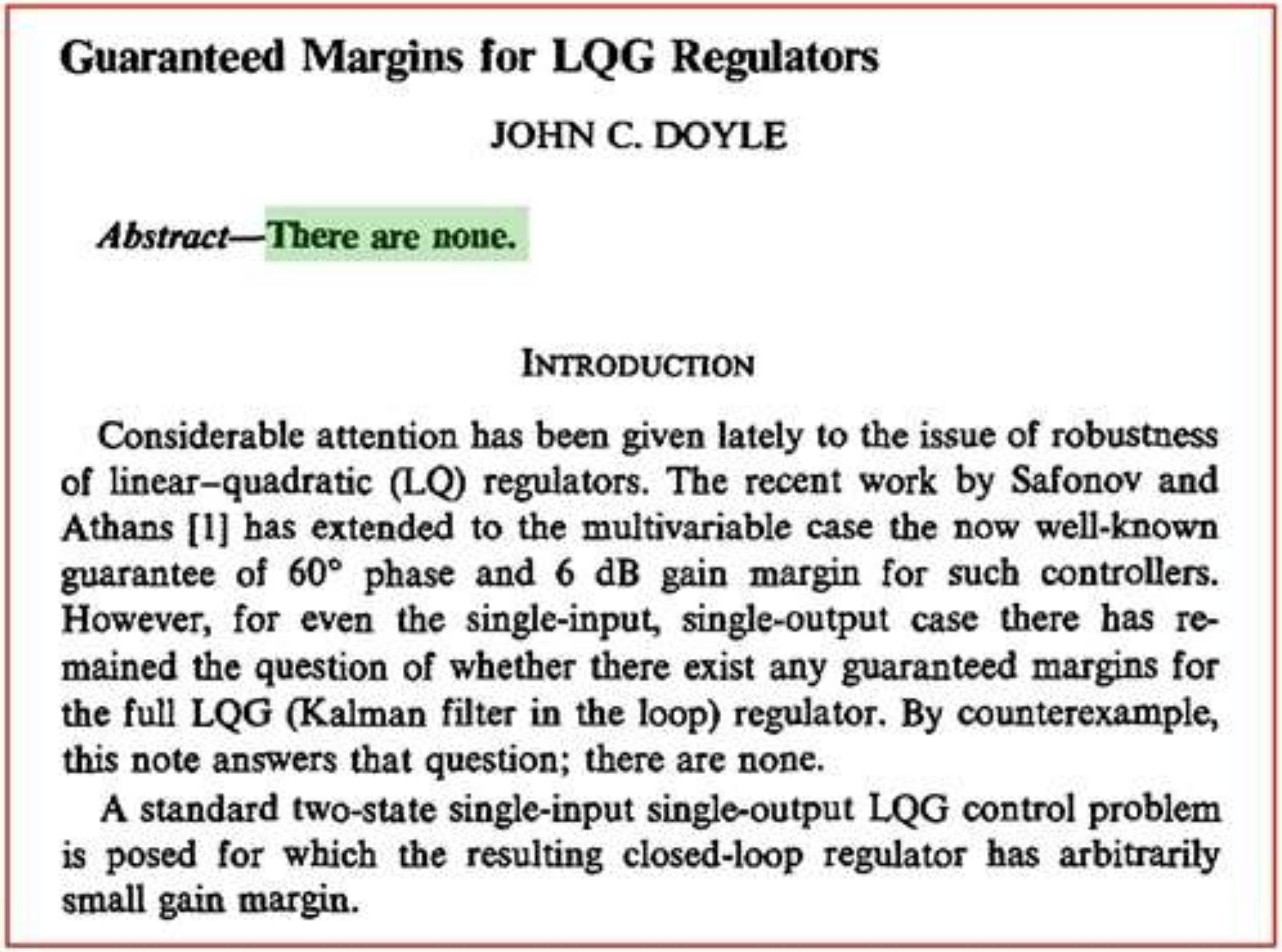
By J. K. Gardner and L. Knopoff

Abstract

Yes.

去除余震后的南加州地震序列呈泊松分布吗?
是的（Yes）

最摘要



Guaranteed Margins for LQG Regulators

JOHN C. DOYLE

Abstract—There are none.

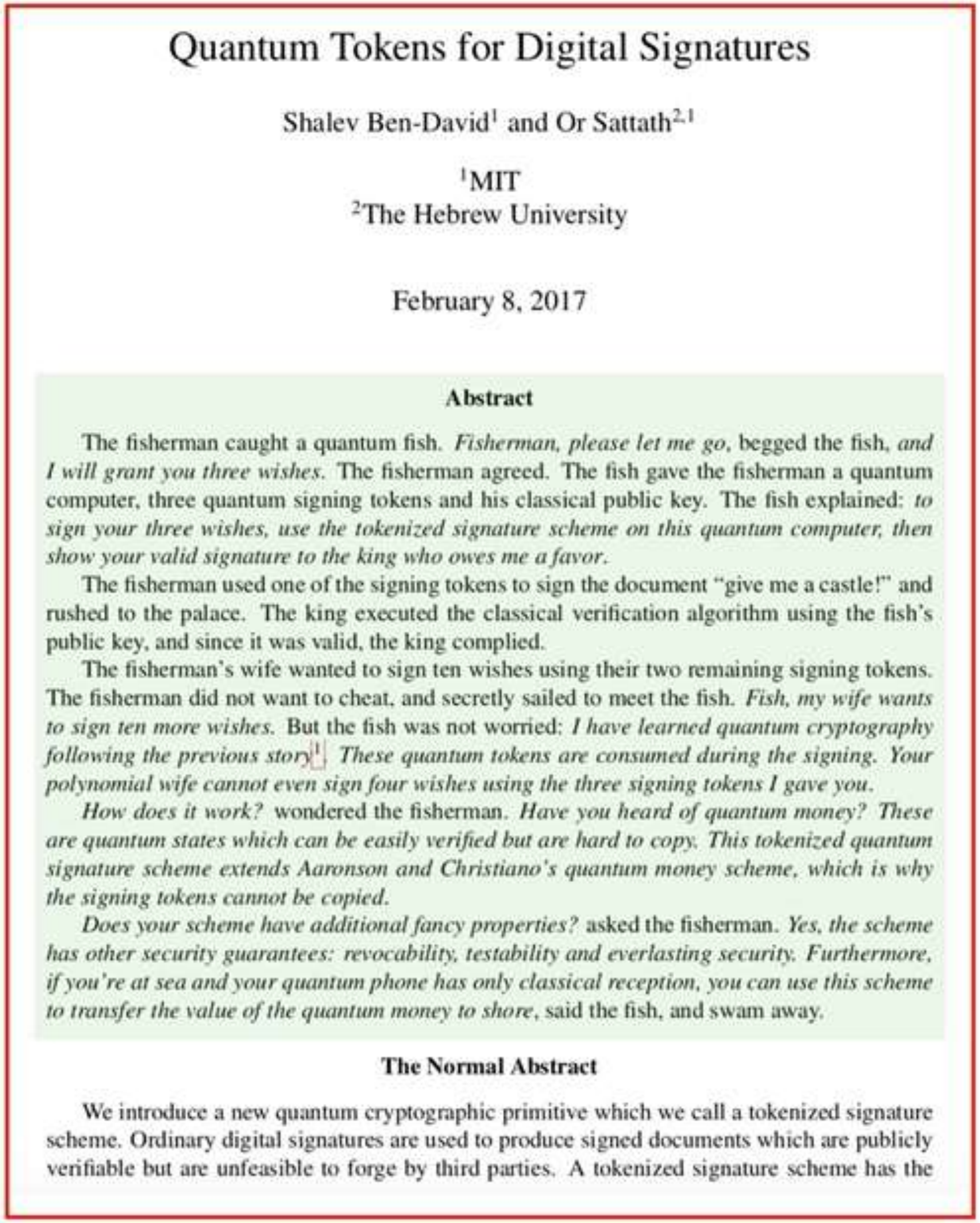
INTRODUCTION

Considerable attention has been given lately to the issue of robustness of linear–quadratic (LQ) regulators. The recent work by Safonov and Athans [1] has extended to the multivariable case the now well-known guarantee of 60° phase and 6 dB gain margin for such controllers. However, for even the single-input, single-output case there has remained the question of whether there exist any guaranteed margins for the full LQG (Kalman filter in the loop) regulator. By counterexample, this note answers that question; there are none.

A standard two-state single-input single-output LQG control problem is posed for which the resulting closed-loop regulator has arbitrarily small gain margin.

LQG调节器的保证裕度
不存在（There are none）

最摘要



Quantum Tokens for Digital Signatures

Shalev Ben-David[1] and Or Sattath[2,1]

[1]MIT
[2]The Hebrew University

February 8, 2017

Abstract

The fisherman caught a quantum fish. *Fisherman, please let me go*, begged the fish, *and I will grant you three wishes*. The fisherman agreed. The fish gave the fisherman a quantum computer, three quantum signing tokens and his classical public key. The fish explained: *to sign your three wishes, use the tokenized signature scheme on this quantum computer, then show your valid signature to the king who owes me a favor*.

The fisherman used one of the signing tokens to sign the document "give me a castle!" and rushed to the palace. The king executed the classical verification algorithm using the fish's public key, and since it was valid, the king complied.

The fisherman's wife wanted to sign ten wishes using their two remaining signing tokens. The fisherman did not want to cheat, and secretly sailed to meet the fish. *Fish, my wife wants to sign ten more wishes*. But the fish was not worried: *I have learned quantum cryptography following the previous story*[1]. *These quantum tokens are consumed during the signing. Your polynomial wife cannot even sign four wishes using the three signing tokens I gave you*.

How does it work? wondered the fisherman. *Have you heard of quantum money? These are quantum states which can be easily verified but are hard to copy. This tokenized quantum signature scheme extends Aaronson and Christiano's quantum money scheme, which is why the signing tokens cannot be copied*.

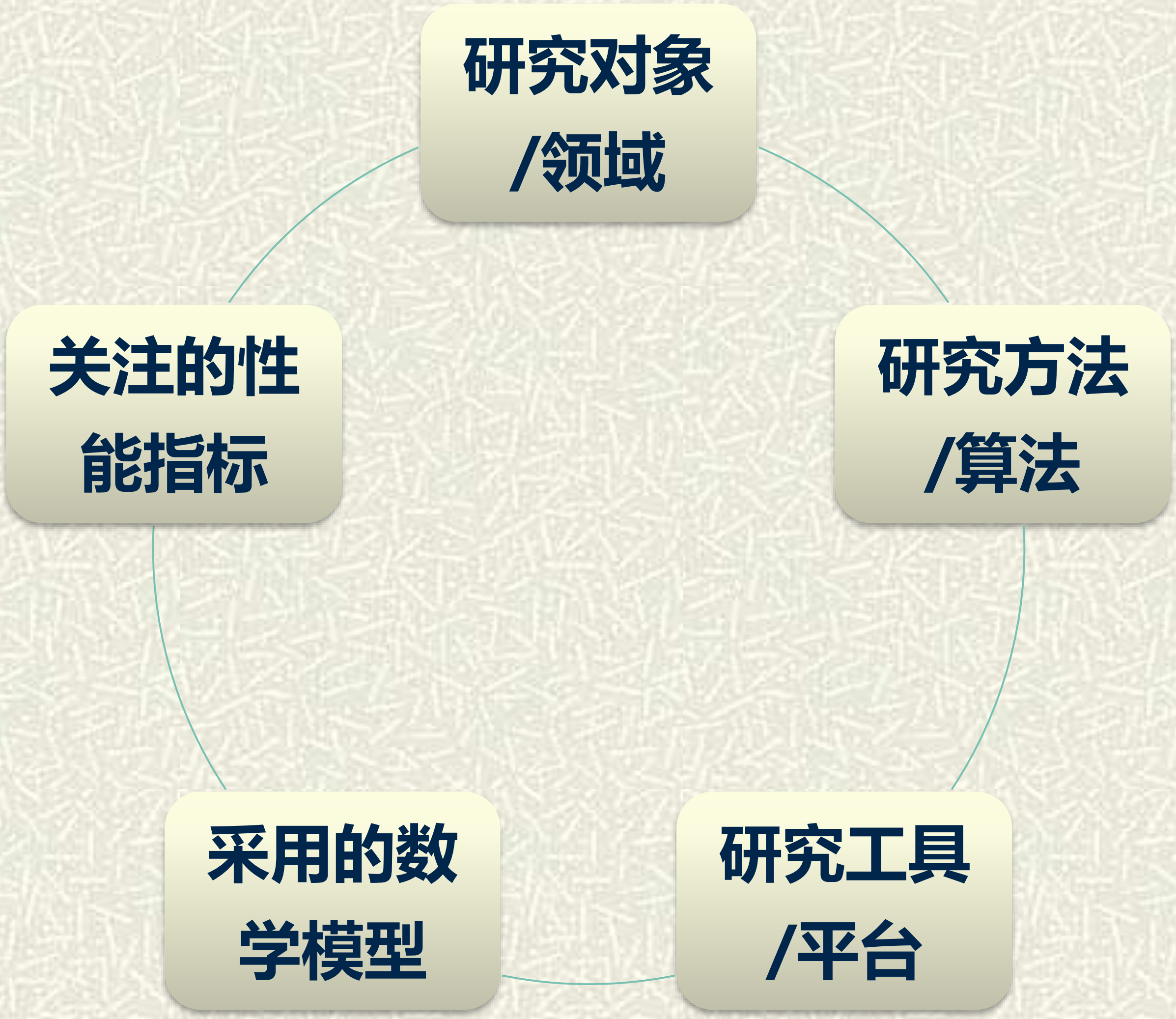
Does your scheme have additional fancy properties? asked the fisherman. *Yes, the scheme has other security guarantees: revocability, testability and everlasting security. Furthermore, if you're at sea and your quantum phone has only classical reception, you can use this scheme to transfer the value of the quantum money to shore*, said the fish, and swam away.

The Normal Abstract

We introduce a new quantum cryptographic primitive which we call a tokenized signature scheme. Ordinary digital signatures are used to produce signed documents which are publicly verifiable but are unfeasible to forge by third parties. A tokenized signature scheme has the

关键词（Keywords）

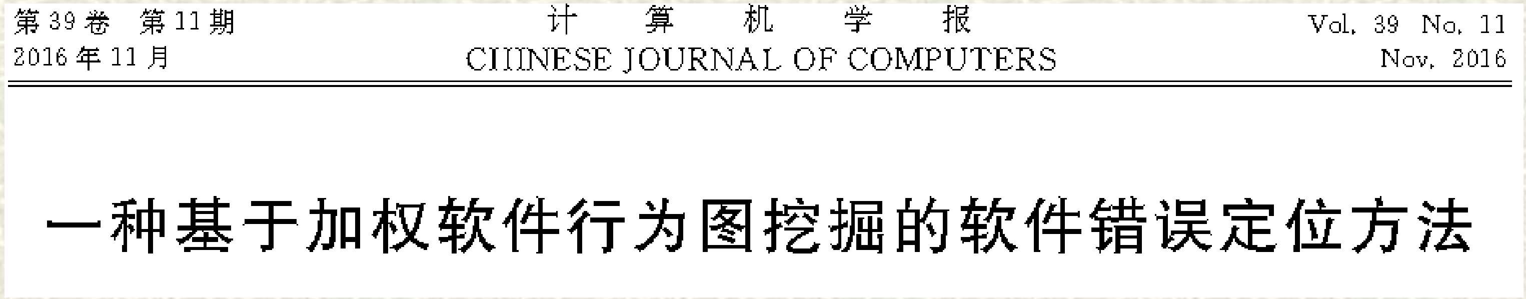
关键词的选择



- 常见问题：不能反映论文的中心议题，不便于检索

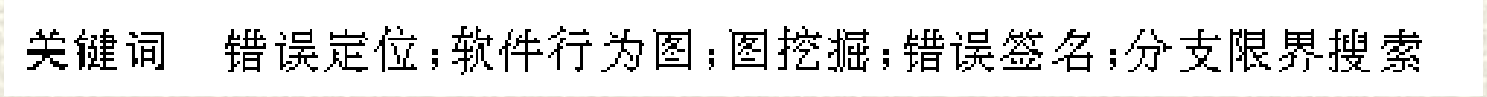
撰写关键词的注意事项

1. 不同关键词之间用“；（分号）”隔开
2. 缩略语应先写中文全称
3. 除专有名词大写外，一律小写
4. 第一个关键词宜选择论文所属的二级学科名（论文分类领域）
5. 关键词的排列顺序，按照关键词的外延从大到小，由前往后排列
6. 论文题目中出现的技术性名词应入选
7. 针对论文有特异性
8. 英文关键词的内容、数量和顺序均应与中文关键词相应



第39卷 第11期 计 算 机 学 报 Vol. 39 No. 11
2016年11月 CHINESE JOURNAL OF COMPUTERS Nov. 2016

一种基于加权软件行为图挖掘的软件错误定位方法



关键词 错误定位；软件行为图；图挖掘；错误签名；分支限界搜索

引言（Introduction）

引言的作用和写作范式

引言的作用和目的

- 作用
  - 强调研究的重要性
  - 提供背景知识铺垫
  - 阐明研究动机
- 目的
  - 吸引读者
  - 看完引言，就能对作者为什么开展这项研究有大致的了解

引言重点写什么?

- What? Why?
  - 研究的背景、理由、贡献，交代本研究的来龙去脉
  - 围绕问题
    - 前人对该问题做了哪些工作, 存在哪些不足
    - 希望解决什么问题
    - 该问题的解决有什么作用和意义
    - 解决问题的理论依据、实验基础和研究方法
    - 预期结果的地位、作用和意义

引言写作的基本范式

- 研究背景（General background of the condition to be discussed)
- 相关领域前人所做的工作和研究概况（What is known so far? A brief description of unknown aspects relevant to the study)
- 有待解决的问题的严重程度和重要性（Primary and secondary questions to be answered by the study)
- 本文的研究目的和贡献（Aim and contribution)
- 论文的组织结构（Organization of the paper)

引言写作的基本范式

- Style
  - Problem X is important
  - A, B, and C have been done
  - A, B, and C have their weakness
  - Our work D
  - Features and advantages of D
  - Results
  - Your contribution
  - Organization of the paper

建立研究背景→

综述前人研究→

挖洞→

填洞

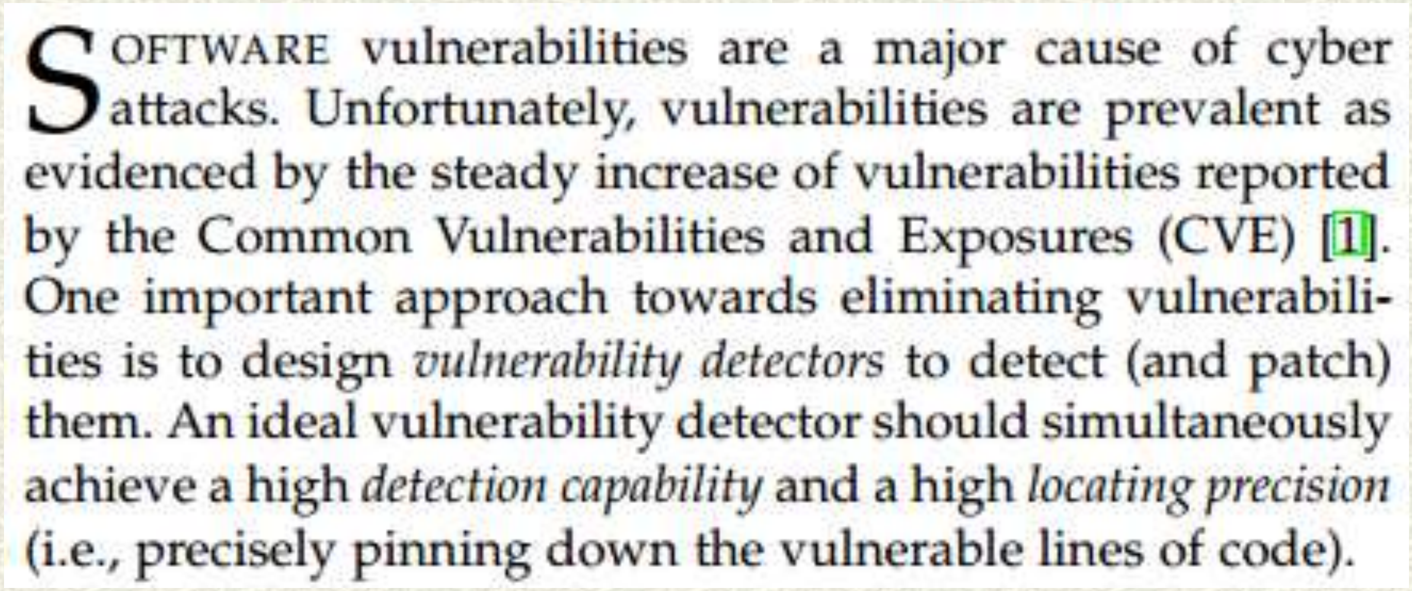
引言写作的指导性问题

1. Background

- What is the context of this problem?
- In what situation or environment can this problem be observed?

2. Rationale

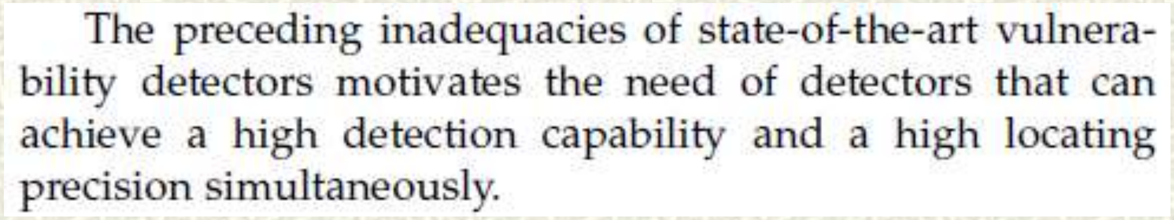
- **Why is this research important?**
- Who will benefit?
- Why do we need to know this?
- **Why does this situation, method, model or piece of equipment need to be improved?**



SOFTWARE vulnerabilities are a major cause of cyber attacks. Unfortunately, vulnerabilities are prevalent as evidenced by the steady increase of vulnerabilities reported by the Common Vulnerabilities and Exposures (CVE) [1]. One important approach towards eliminating vulnerabilities is to design *vulnerability detectors* to detect (and patch) them. An ideal vulnerability detector should simultaneously achieve a high *detection capability* and a high *locating precision* (i.e., precisely pinning down the vulnerable lines of code).

引言写作的指导性问题

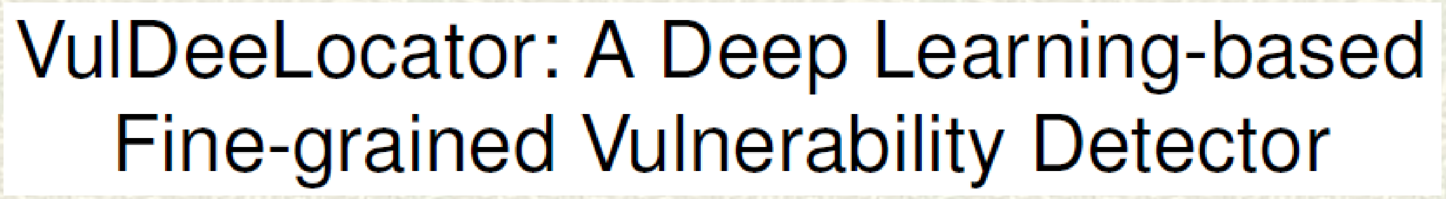
- **3. Problem Statement**
  - What is it we don't know?
  - What is the gap in our knowledge this research will fill?
  - What needs to be improved?
- **4. Objectives**
  - What steps will the researcher take to try and fill this gap or improve the situation?
- **5. Scope**
  - Is there any aspect of the problem the researcher will not discuss?
  - Is the study limited to the situation?
- **6. Limitations**
  - Is there any factor, condition or circumstance that prevents the researcher from achieving all his/her objectives?


The preceding inadequacies of state-of-the-art vulnerability detectors motivates the need of detectors that can achieve a high detection capability and a high locating precision simultaneously.

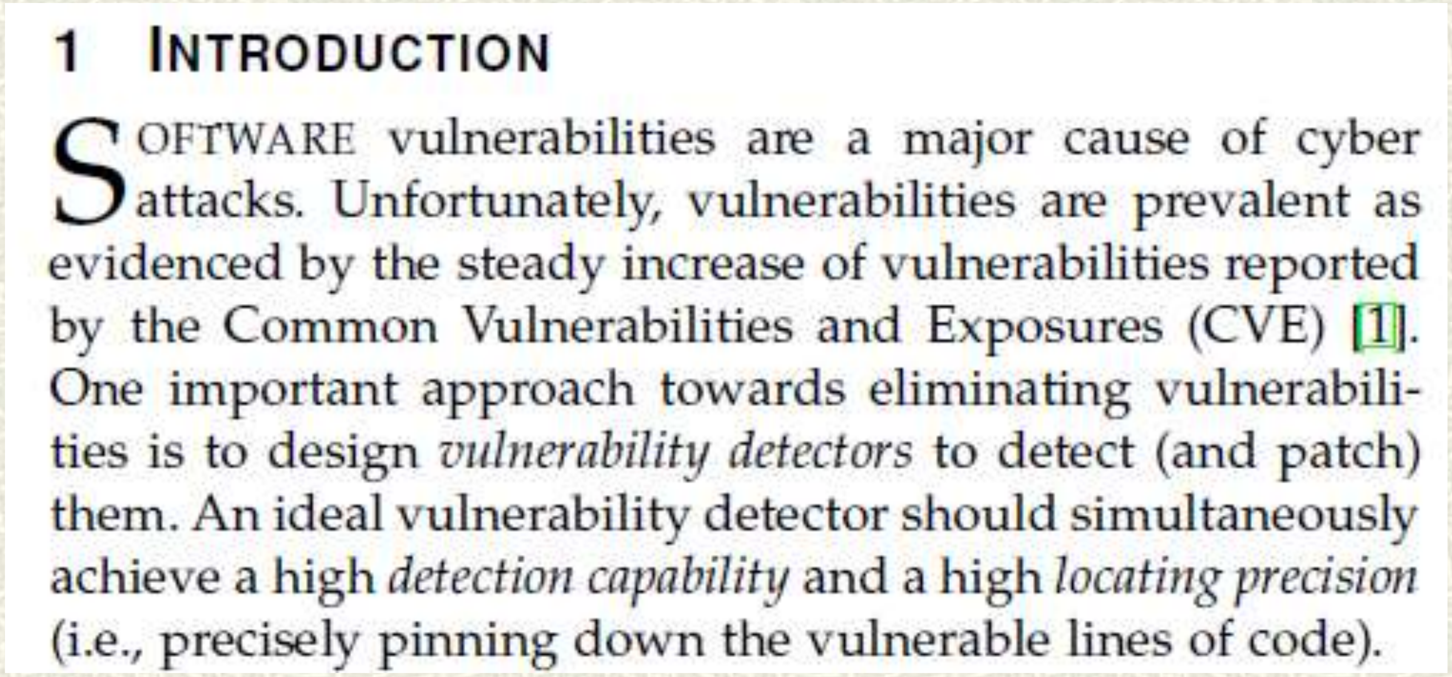
引言写作的基本原则

引言写作的基本原则

- **1.开门见山点题，不绕圈子**
  - **点到题目中的关键词**



VulDeeLocator: A Deep Learning-based Fine-grained Vulnerability Detector

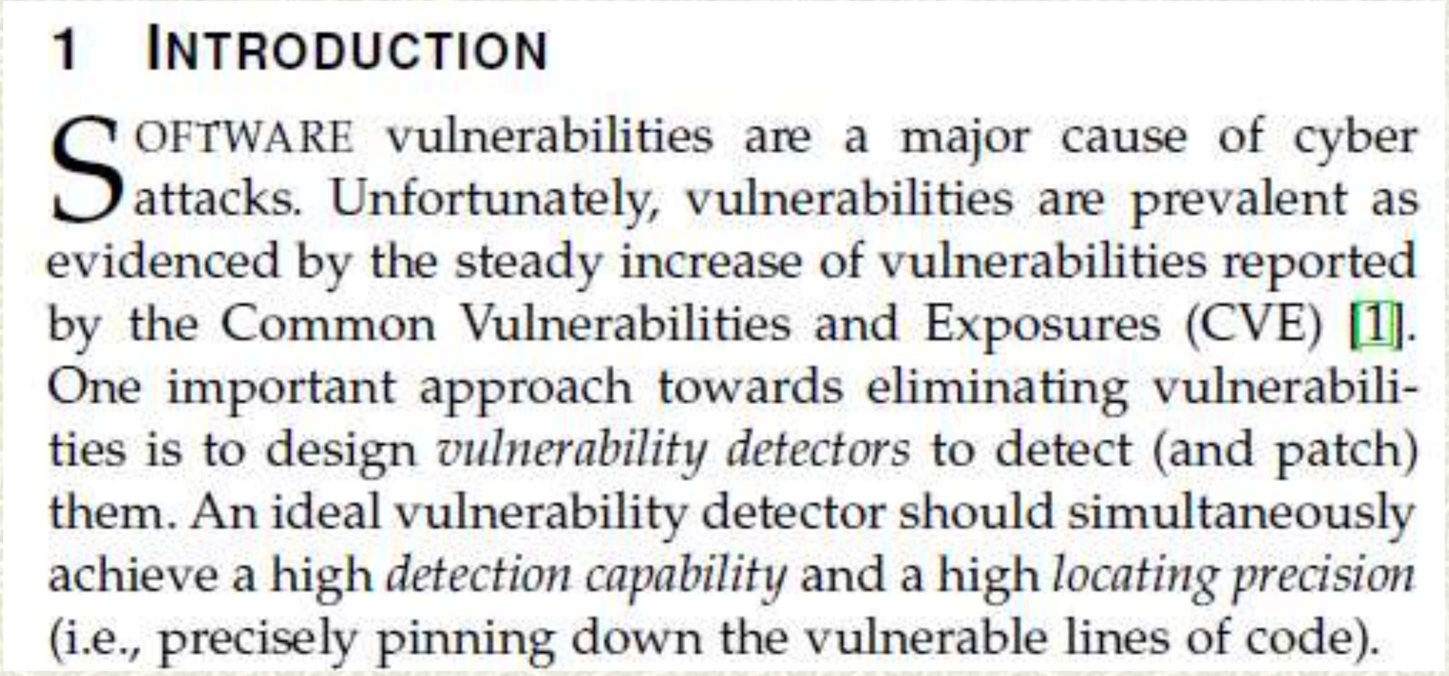


1 INTRODUCTION

SOFTWARE vulnerabilities are a major cause of cyber attacks. Unfortunately, vulnerabilities are prevalent as evidenced by the steady increase of vulnerabilities reported by the Common Vulnerabilities and Exposures (CVE) [1]. One important approach towards eliminating vulnerabilities is to design *vulnerability detectors* to detect (and patch) them. An ideal vulnerability detector should simultaneously achieve a high *detection capability* and a high *locating precision* (i.e., precisely pinning down the vulnerable lines of code).

引言写作的基本原则

- **2.背景介绍要言简意赅，简洁清晰，切忌长篇大论**
  - **Introduction to the background should be concise and clear**
  - **Never write on extensive review of the field**
  - **Never make this section into a history lesson**



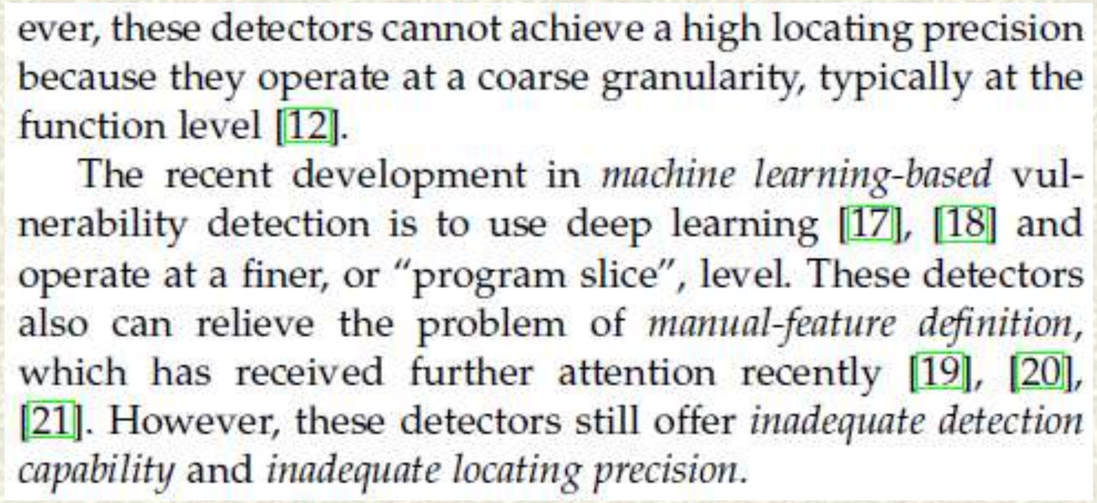
1 INTRODUCTION

SOFTWARE vulnerabilities are a major cause of cyber attacks. Unfortunately, vulnerabilities are prevalent as evidenced by the steady increase of vulnerabilities reported by the Common Vulnerabilities and Exposures (CVE) [1]. One important approach towards eliminating vulnerabilities is to design *vulnerability detectors* to detect (and patch) them. An ideal vulnerability detector should simultaneously achieve a high *detection capability* and a high *locating precision* (i.e., precisely pinning down the vulnerable lines of code).

引言写作的基本原则

3.突出问题的重要性

- Highlight the importance of your study
  - Includes statistics to establish the importance of a problem area
  - show that prominent people or influential authors have considered and addressed the issue that is being researched
  - show that a topic is of current interest



ever, these detectors cannot achieve a high locating precision because they operate at a coarse granularity, typically at the function level [12].

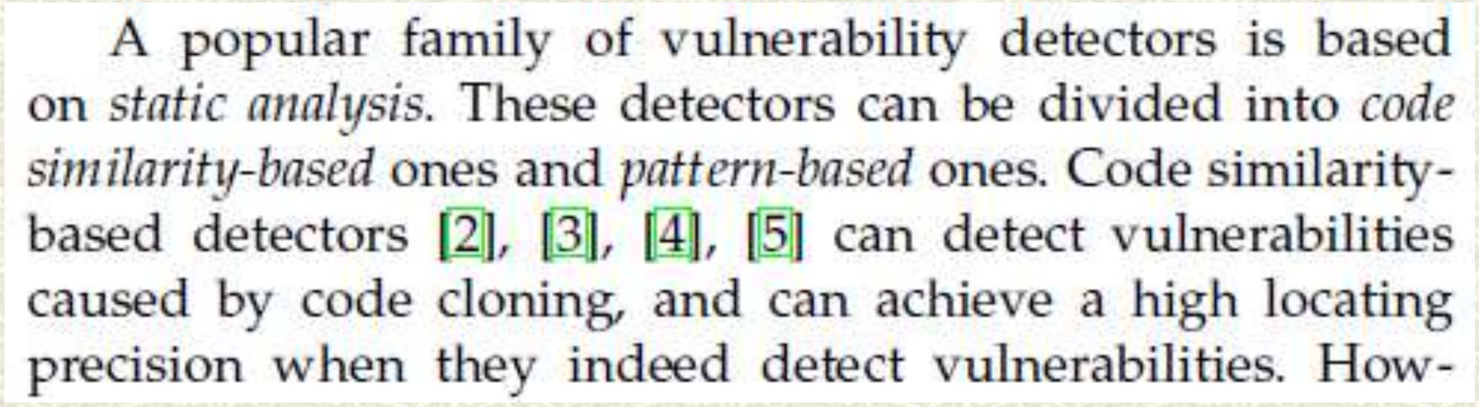
The recent development in *machine learning-based* vulnerability detection is to use deep learning [17], [18] and operate at a finer, or "program slice", level. These detectors also can relieve the problem of *manual-feature definition*, which has received further attention recently [19], [20], [21]. However, these detectors still offer *inadequate detection capability* and *inadequate locating precision*.

引言写作的基本原则

- 4.Citing relevant literature
  - For the purpose of leading the reader to some point, it is necessary to review the relevant literature, but you should provide the most salient background rather than an exhaustive review
  - 切忌写成讨论、综述和回顾
  - Ensure that the literature cited is balanced up to date and relevant
  - Avoid general reference works such as textbooks
  - Pay attention to the pertinent literature published while the study is in progress

引言写作的基本原则

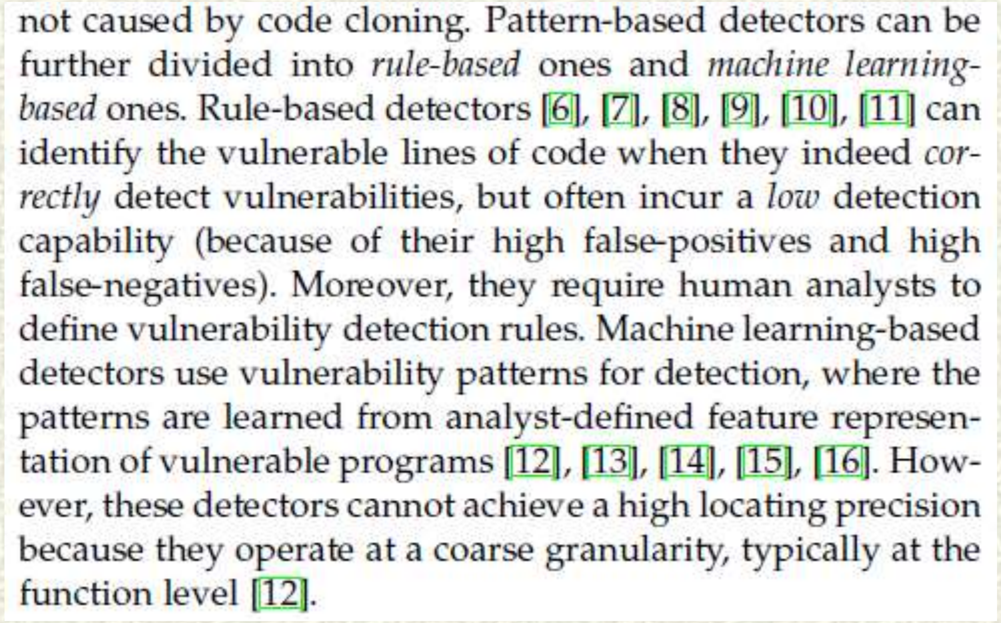
- 4.Citing relevant literature
  - Literature should be organized around and related directly to the thesis or research question you are developing
  - 段首主题句概括，Topic sentence



A popular family of vulnerability detectors is based on *static analysis*. These detectors can be divided into *code similarity-based* ones and *pattern-based* ones. Code similarity-based detectors [2], [3], [4], [5] can detect vulnerabilities caused by code cloning, and can achieve a high locating precision when they indeed detect vulnerabilities. How-

引言写作的基本原则

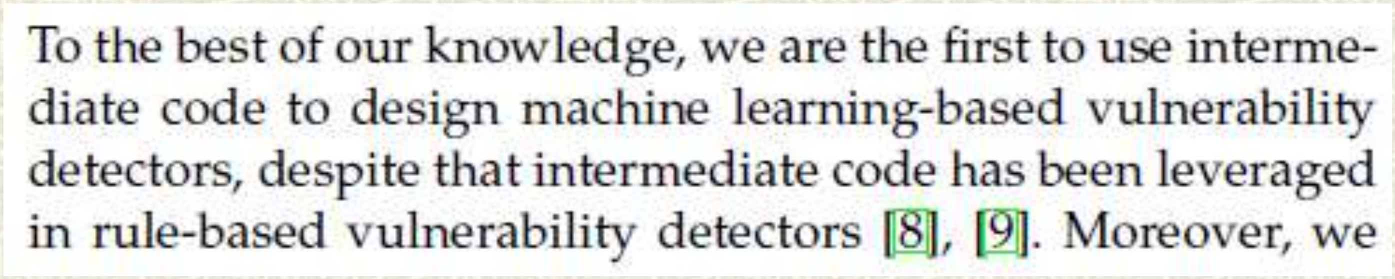
- 5.如实评述，防止吹嘘自己和贬低别人
  - Modestly emphasize your contributions
    - You may explain how the present experiment will help to clarify or expand the knowledge in this general area. Emphasize your specific contribution to the topic.
    - However, never minimize or dismiss contributions made by others.


not caused by code cloning. Pattern-based detectors can be further divided into *rule-based* ones and *machine learning-based* ones. Rule-based detectors [6], [7], [8], [9], [10], [11] can identify the vulnerable lines of code when they indeed *correctly* detect vulnerabilities, but often incur a *low* detection capability (because of their high false-positives and high false-negatives). Moreover, they require human analysts to define vulnerability detection rules. Machine learning-based detectors use vulnerability patterns for detection, where the patterns are learned from analyst-defined feature representation of vulnerable programs [12], [13], [14], [15], [16]. However, these detectors cannot achieve a high locating precision because they operate at a coarse granularity, typically at the function level [12].

引言写作的基本原则

6.杜绝大话，用词谨慎

- “填补空白”、“达到先进水平”、“前人从未研究过”等
- Expressions such as "novel", "first time", "first ever", are not preferred.
- The paper expands on the work reported elsewhere for the UK city of Lecester[12] and is complementary to the analysis of the CaRB data presented in Kelly er al.[13]. Thus，to the authors' knowledge, one of the first large national scale studies of summertime temperatures and thermal comfort in English homes has been undertaken.
- “以往相关报道较为少见”，“据作者所知前人很少涉及……”



To the best of our knowledge, we are the first to use intermediate code to design machine learning-based vulnerability detectors, despite that intermediate code has been leveraged in rule-based vulnerability detectors [8], [9]. Moreover, we

引言写作的基本原则

- 7.话别说太满
  - Avoid making claims that are too bold, like
    - "this is a potential cure for all cancer “
- 8.内容不要与摘要雷同
- 9.引言与结论呼应，在引言中提出的问题，在结论中解答
- 10.不要插图列表和进行数学公式的推导证明

引言写作的常见问题

引言撰写中的常见问题

- 1）引言过短
- 2）重要文献缺失
- 3）简单罗列，引文间没有太多逻辑，没有支撑作用
- 4）引言中同一位置引用文献过多
  - “……工作[1-20]”
- 5）引文中否定前人工作

引言撰写中的常见问题

- 6）试图回顾有关该主题的所有可用证据
- 7）引言部分提到的问题，没有在讨论部分回答
- 8）引言内容与摘要雷同，像是摘要的注释
- 9）出现了插图列表和数学公式的推导证明

问题引言示例

- 在对砌体墙板各种荷载工况下的破坏情况，即开裂模式与承载能力的分析中，由于砌体墙板材料性质以及制作等带来的高度变异性，**致使砌体墙板工作机理的精确分析十分困难**[1-8]。目前，砌体墙板分析广泛应用的是有限元法[1-3,5,8]，但是近30年的研究表明：无论砌体结构材料性质的本构关系多么精细，只要砌体结构全局材料性质参数一致，有限元方法在许多情况下仍然难以精确预测砌体墙板开裂模式与破坏荷载。因此，研究人员尝试了在砌体墙板有限元分析中融入材料性质的变异性，来改进分析的精度，例如：1991年Lawrence等[1]在砌体墙板上各个区域引入随机强度干扰系数来反映砌体性质的变异，2003年Rafiq等[6]以砌体墙板试验数据为基础获取墙板区域刚度修正系数，在一定程度上使有限元模拟结果更靠近了实际。然而，由于有限元理想化假设的限制，其分析结果与砌体墙板的试验结果相比仍然误差很大，甚至失

避免这种笼统而模糊的集中引用，选择代表性文献标引

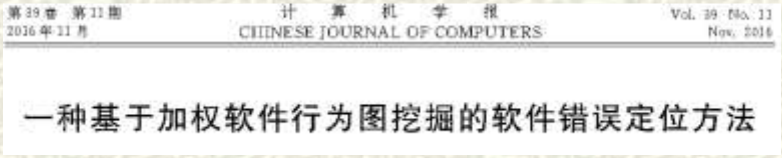
问题引言示例

- 人工智能技术的发展为精确分析砌体墙板开辟了新途径。2006年Zhou等[9]发展了一种细胞自动机方法，即以标准试验墙板开裂模式为基础预测新墙板开裂模式的方法。2010年Zhang等[10] 应用BP神经网络技术，建立了预测砌体小墙开裂模式的方法。
- 近年来，支持向量机在土木工程领域的应用越来越广泛，譬如在结构及构件损伤识别[11]、结构可靠性分析[12]、岩土工程（如结构岩爆识别模型[13],边坡稳定[14]等）、结构健康监测[15]等的应用，发挥了其解决小样本、非线性及高位模式识别的特有优势。本文尝试应用支持向量机对砌体墙板开裂模式进行数值建模，并建立相应的预测方法，意在使支持向量机在砌体墙板开裂模式预测上得以应用，探索一种在一定程度更为精确、简单易行的预测方法；同时，探索现有墙板试验记录的进一步应用，使之融入结构分析方法中。

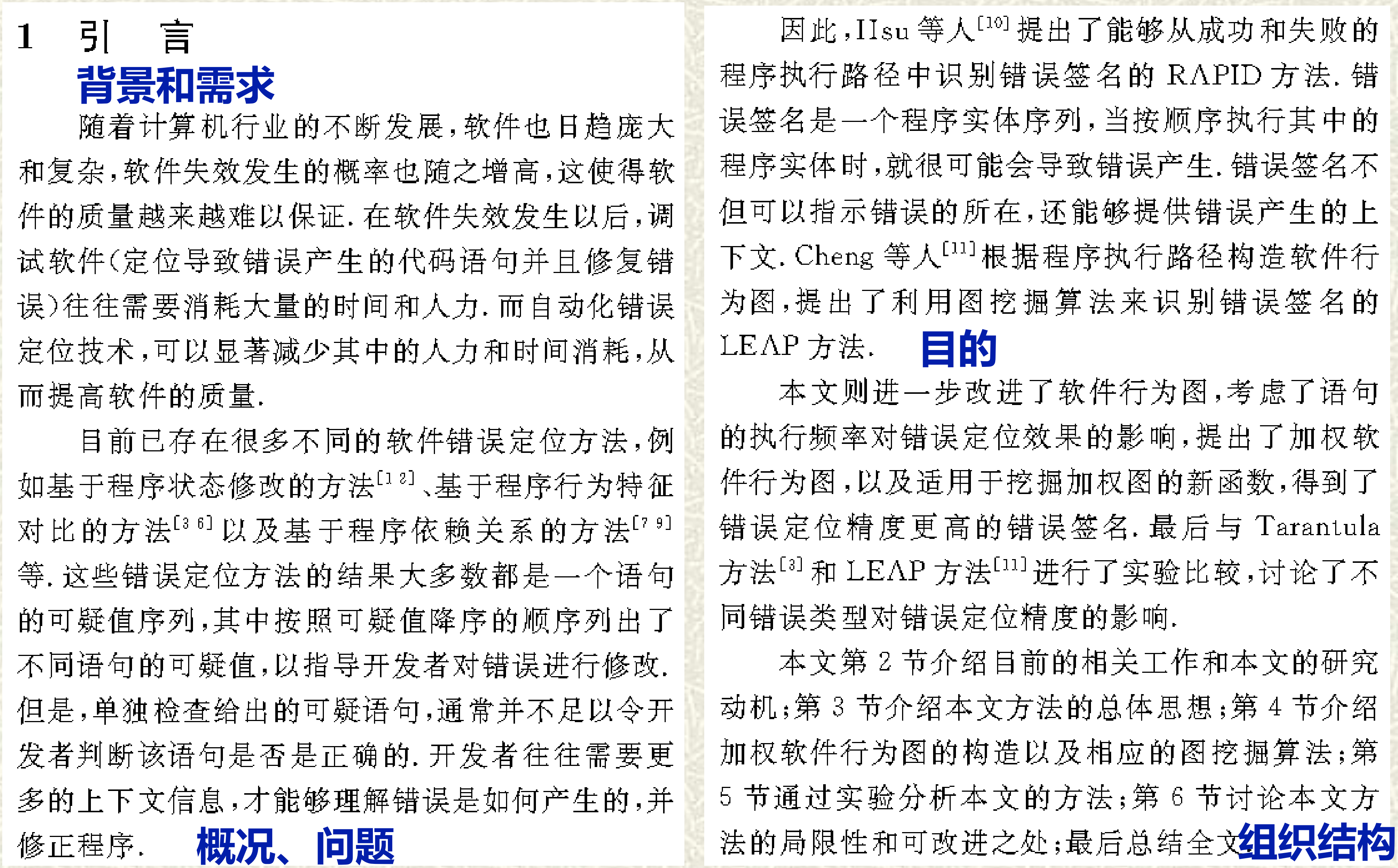
建议删除这句，这些文献不宜标引，
不用隆重介绍研究方法在其他领域的应用，
对本文研究没有支撑论证作用

中英文引言示例与对比

中文引言示例


第39卷 第11期 2016年11月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 39 No. 11 Nov. 2016

一种基于加权软件行为图挖掘的软件错误定位方法


1 引 言

背景和需求

随着计算机行业的不断发展，软件也日趋庞大和复杂，软件失效发生的概率也随之增高，这使得软件的质量越来越难以保证. 在软件失效发生以后，调试软件（定位导致错误产生的代码语句并且修复错误）往往需要消耗大量的时间和人力. 而自动化错误定位技术，可以显著减少其中的人力和时间消耗，从而提高软件的质量.

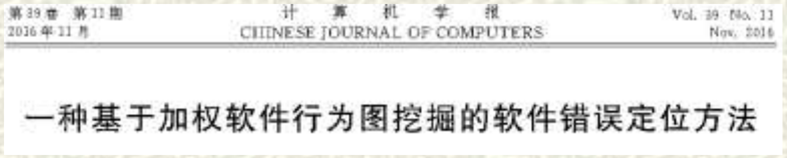
目前已存在很多不同的软件错误定位方法，例如基于程序状态修改的方法[1-2]、基于程序行为特征对比的方法[3-6]以及基于程序依赖关系的方法[7-9]等. 这些错误定位方法的结果大多数都是一个语句的可疑值序列，其中按照可疑值降序的顺序列出了不同语句的可疑值，以指导开发者对错误进行修改. 但是，单独检查给出的可疑语句，通常并不足以令开发者判断该语句是否是正确的. 开发者往往需要更多的上下文信息，才能够理解错误是如何产生的，并修正程序. **概况、问题**

因此，Hsu 等人[10]提出了能够从成功和失败的程序执行路径中识别错误签名的 RAPID 方法. 错误签名是一个程序实体序列，当按顺序执行其中的程序实体时，就很可能会导致错误产生. 错误签名不但可以指示错误的所在，还能够提供错误产生的上下文. Cheng 等人[11]根据程序执行路径构造软件行为图，提出了利用图挖掘算法来识别错误签名的 LEAP 方法. **目的**

本文则进一步改进了软件行为图，考虑了语句的执行频率对错误定位效果的影响，提出了加权软件行为图，以及适用于挖掘加权图的新函数，得到了错误定位精度更高的错误签名. 最后与 Tarantula 方法[3]和 LEAP 方法[11]进行了实验比较，讨论了不同错误类型对错误定位精度的影响.

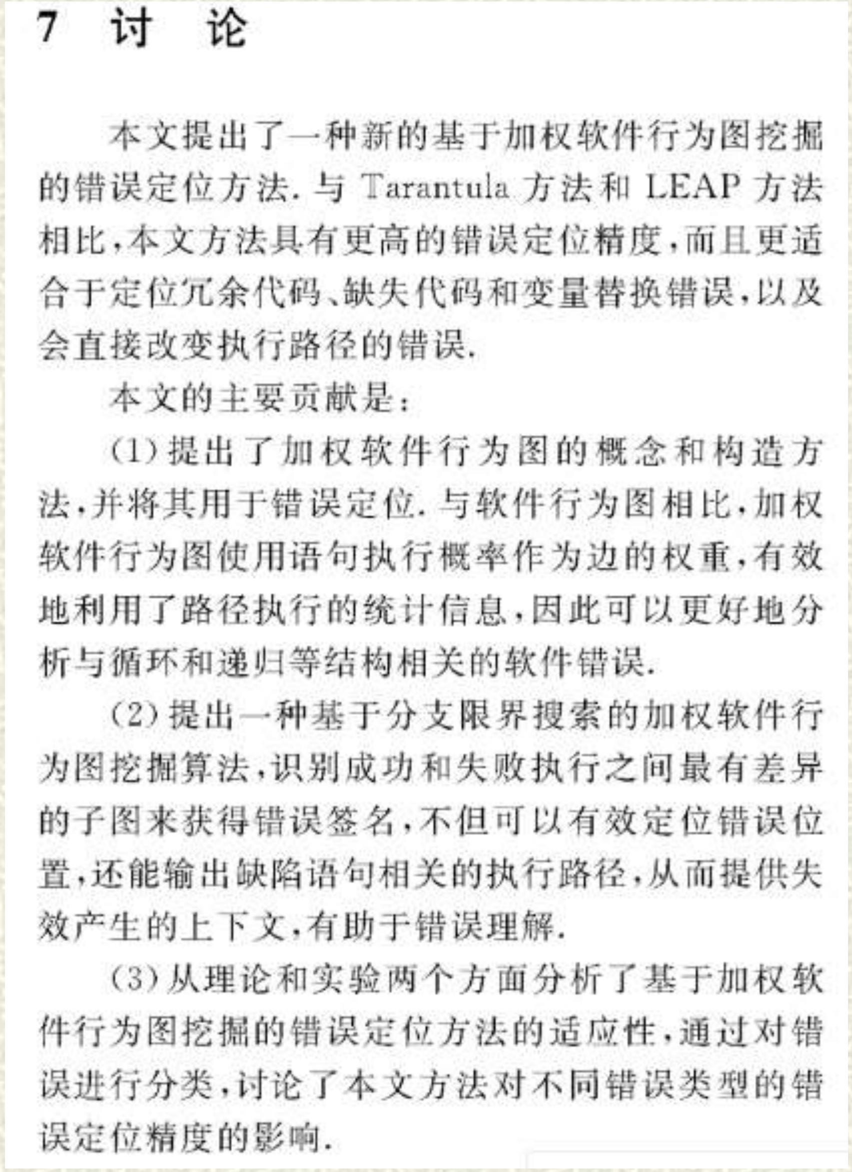
本文第 2 节介绍目前的相关工作和本文的研究动机；第 3 节介绍本文方法的总体思想；第 4 节介绍加权软件行为图的构造以及相应的图挖掘算法；第 5 节通过实验分析本文的方法；第 6 节讨论本文方法的局限性和可改进之处；最后总结全文. **组织结构**

中文引言示例


第39卷 第11期 2016年11月 计 算 机 学 报 CHINESE JOURNAL OF COMPUTERS Vol. 39 No. 11 Nov. 2016

一种基于加权软件行为图挖掘的软件错误定位方法

主要贡献没有放在引言中


7 讨 论

本文提出了一种新的基于加权软件行为图挖掘的错误定位方法. 与 Tarantula 方法和 LEAP 方法相比,本文方法具有更高的错误定位精度,而且更适合于定位冗余代码、缺失代码和变量替换错误,以及会直接改变执行路径的错误.

本文的主要贡献是:

(1) 提出了加权软件行为图的概念和构造方法,并将其用于错误定位. 与软件行为图相比,加权软件行为图使用语句执行概率作为边的权重,有效地利用了路径执行的统计信息,因此可以更好地分析与循环和递归等结构相关的软件错误.

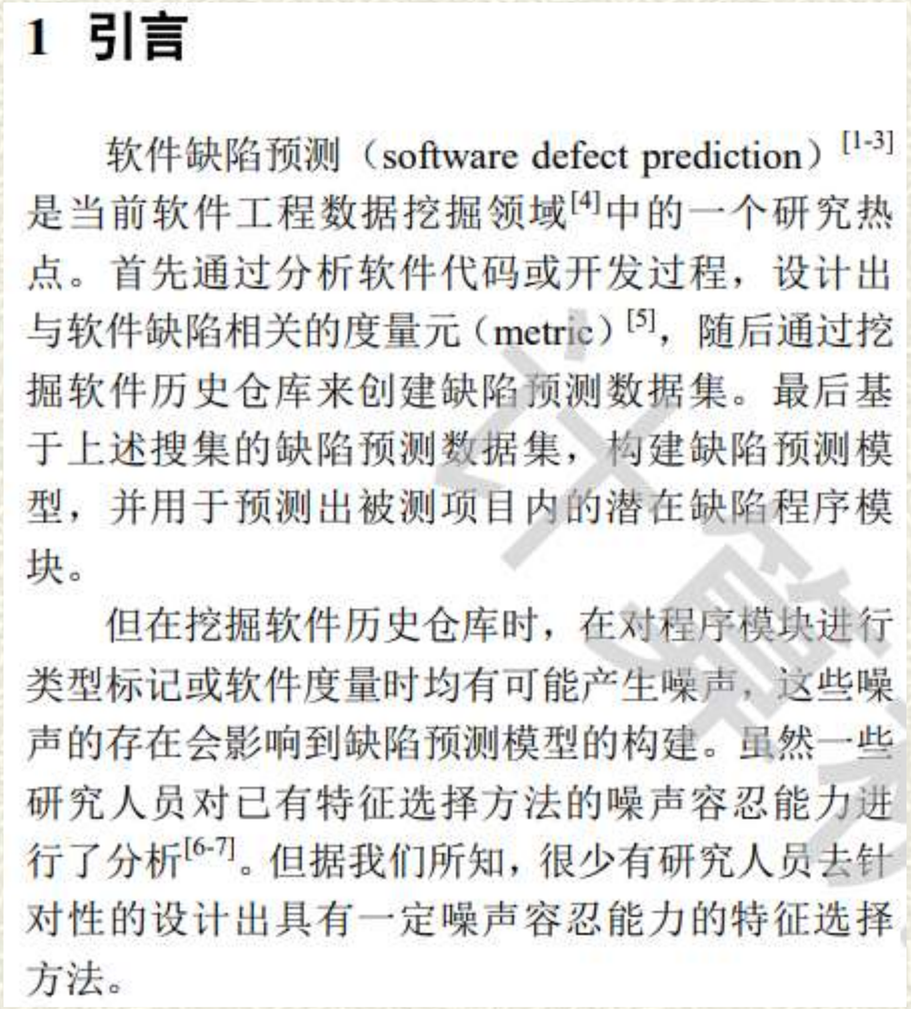
(2) 提出一种基于分支限界搜索的加权软件行为图挖掘算法,识别成功和失败执行之间最有差异的子图来获得错误签名,不但可以有效定位错误位置,还能输出缺陷语句相关的执行路径,从而提供失效产生的上下文,有助于错误理解.

(3) 从理论和实验两个方面分析了基于加权软件行为图挖掘的错误定位方法的适应性,通过对错误进行分类,讨论了本文方法对不同错误类型的错误定位精度的影响.

中文引言示例

一种面向软件缺陷预测的可容忍噪声的

特征选择框架

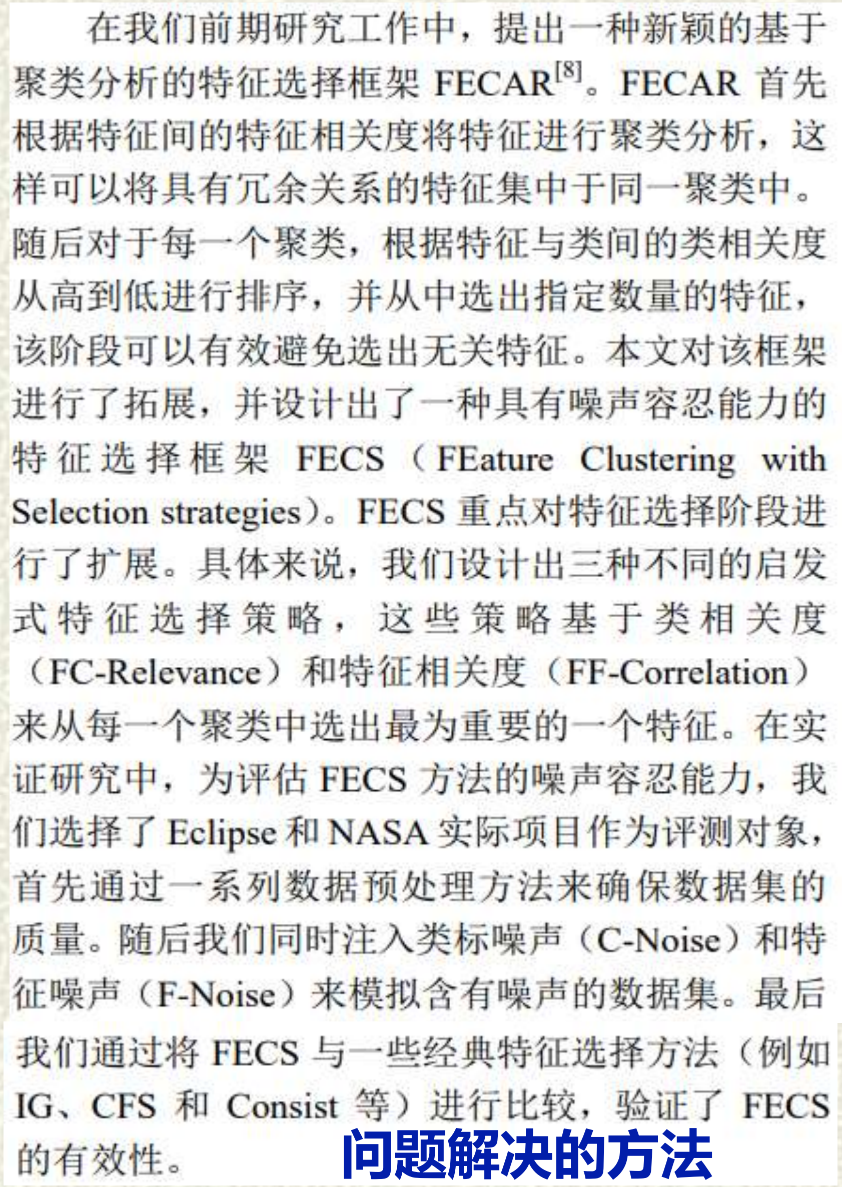


1 引言

软件缺陷预测（software defect prediction）[1-3]是当前软件工程数据挖掘领域[4]中的一个研究热点。首先通过分析软件代码或开发过程，设计出与软件缺陷相关的度量元（metric）[5]，随后通过挖掘软件历史仓库来创建缺陷预测数据集。最后基于上述搜集的缺陷预测数据集，构建缺陷预测模型，并用于预测出被测项目内的潜在缺陷程序模块。

但在挖掘软件历史仓库时，在对程序模块进行类型标记或软件度量时均有可能产生噪声，这些噪声的存在会影响到缺陷预测模型的构建。虽然一些研究人员对已有特征选择方法的噪声容忍能力进行了分析[6-7]。但据我们所知，很少有研究人员去针对性的设计出具有一定噪声容忍能力的特征选择方法。

背景和问题

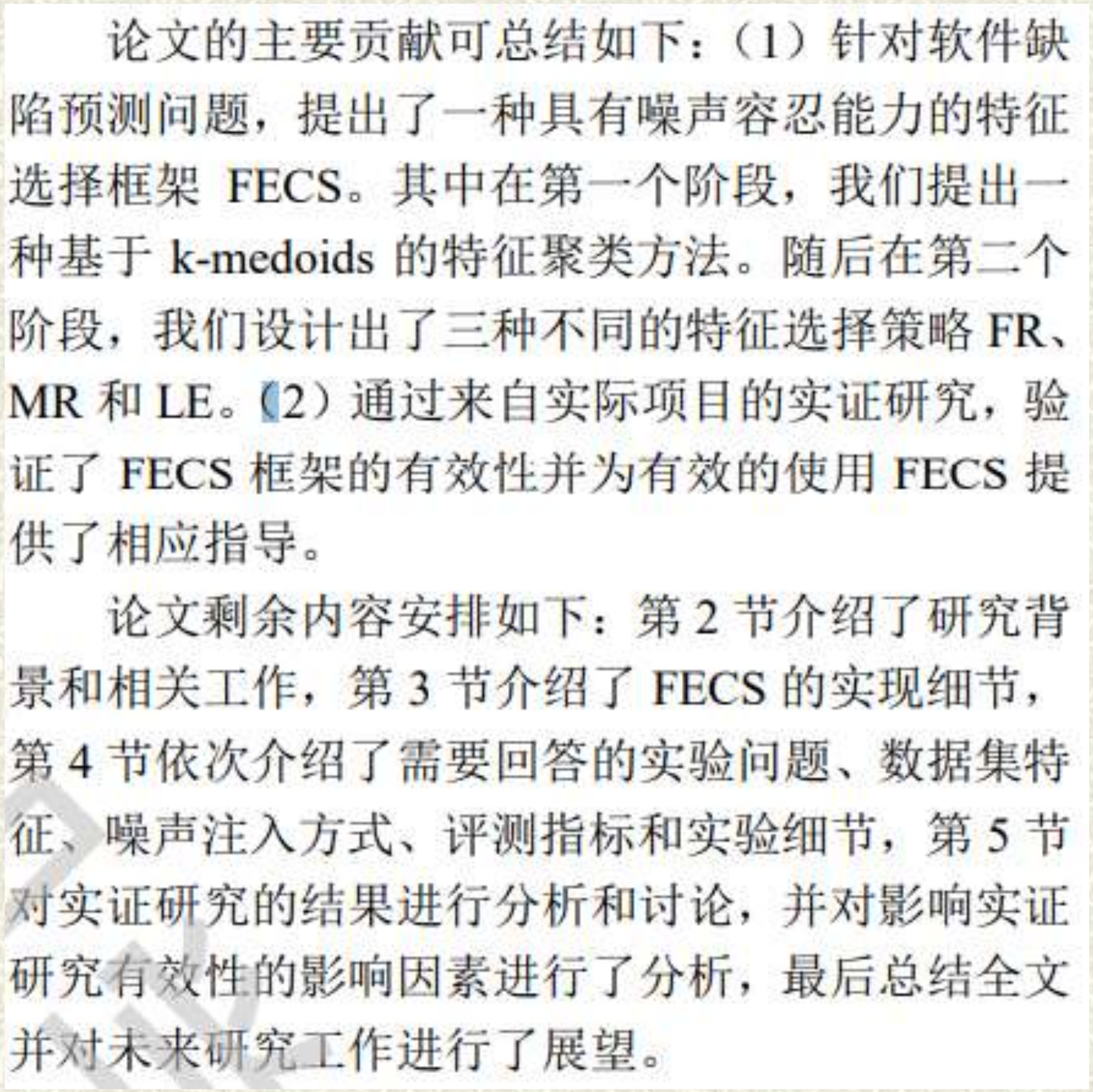


在我们前期研究工作中，提出一种新颖的基于聚类分析的特征选择框架 FECAR[8]。FECAR 首先根据特征间的特征相关度将特征进行聚类分析，这样可以将具有冗余关系的特征集中于同一聚类中。随后对于每一个聚类，根据特征与类间的类相关度从高到低进行排序，并从中选出指定数量的特征，该阶段可以有效避免选出无关特征。本文对该框架进行了拓展，并设计出了一种具有噪声容忍能力的特征选择框架 FECS（FEature Clustering with Selection strategies）。FECS 重点对特征选择阶段进行了扩展。具体来说，我们设计出三种不同的启发式特征选择策略，这些策略基于类相关度（FC-Relevance）和特征相关度（FF-Correlation）来从每一个聚类中选出最为重要的一个特征。在实证研究中，为评估 FECS 方法的噪声容忍能力，我们选择了 Eclipse 和 NASA 实际项目作为评测对象，首先通过一系列数据预处理方法来确保数据集的质量。随后我们同时注入类标噪声（C-Noise）和特征噪声（F-Noise）来模拟含有噪声的数据集。最后我们通过将 FECS 与一些经典特征选择方法（例如 IG、CFS 和 Consist 等）进行比较，验证了 FECS 的有效性。

问题解决的方法

中文引言示例

一种面向软件缺陷预测的可容忍噪声的特征选择框架



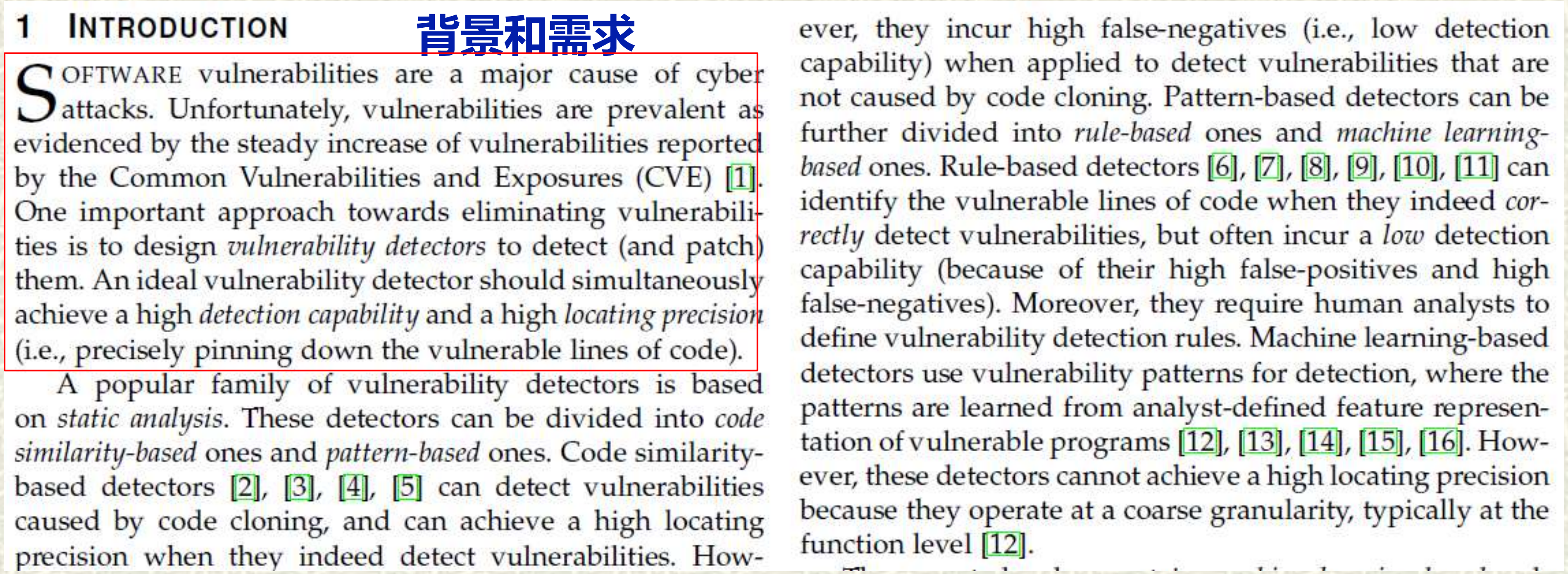
论文的主要贡献可总结如下：（1）针对软件缺陷预测问题，提出了一种具有噪声容忍能力的特征选择框架 FECS。其中在第一个阶段，我们提出一种基于 k-medoids 的特征聚类方法。随后在第二个阶段，我们设计出了三种不同的特征选择策略 FR、MR 和 LE。（2）通过来自实际项目的实证研究，验证了 FECS 框架的有效性并为有效的使用 FECS 提供了相应指导。

论文剩余内容安排如下：第 2 节介绍了研究背景和相关工作，第 3 节介绍了 FECS 的实现细节，第 4 节依次介绍了需要回答的实验问题、数据集特征、噪声注入方式、评测指标和实验细节，第 5 节对实证研究的结果进行分析和讨论，并对影响实证研究有效性的影响因素进行了分析，最后总结全文并对未来研究工作进行了展望。

主要贡献
组织结构

英文引言示例

VulDeeLocator: A Deep Learning-based Fine-grained Vulnerability Detector

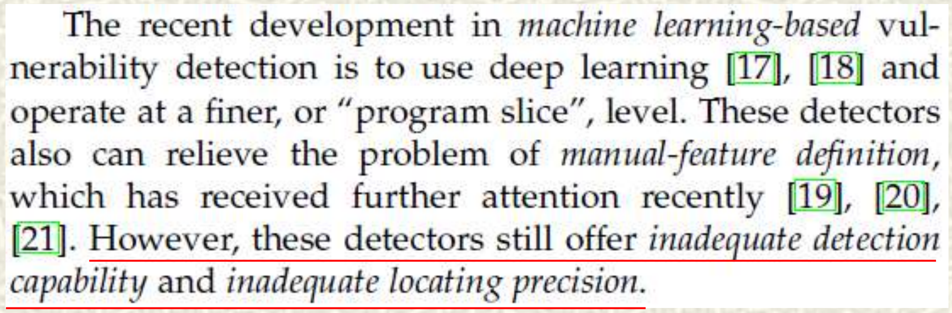


1 INTRODUCTION

背景和需求

SOFTWARE vulnerabilities are a major cause of cyber attacks. Unfortunately, vulnerabilities are prevalent as evidenced by the steady increase of vulnerabilities reported by the Common Vulnerabilities and Exposures (CVE) [1]. One important approach towards eliminating vulnerabilities is to design *vulnerability detectors* to detect (and patch) them. An ideal vulnerability detector should simultaneously achieve a high *detection capability* and a high *locating precision* (i.e., precisely pinning down the vulnerable lines of code).

A popular family of vulnerability detectors is based on *static analysis*. These detectors can be divided into *code similarity-based* ones and *pattern-based* ones. Code similarity-based detectors [2], [3], [4], [5] can detect vulnerabilities caused by code cloning, and can achieve a high locating precision when they indeed detect vulnerabilities. However, they incur high false-negatives (i.e., low detection capability) when applied to detect vulnerabilities that are not caused by code cloning. Pattern-based detectors can be further divided into *rule-based* ones and *machine learning-based* ones. Rule-based detectors [6], [7], [8], [9], [10], [11] can identify the vulnerable lines of code when they indeed *correctly* detect vulnerabilities, but often incur a *low* detection capability (because of their high false-positives and high false-negatives). Moreover, they require human analysts to define vulnerability detection rules. Machine learning-based detectors use vulnerability patterns for detection, where the patterns are learned from analyst-defined feature representation of vulnerable programs [12], [13], [14], [15], [16]. However, these detectors cannot achieve a high locating precision because they operate at a coarse granularity, typically at the function level [12].

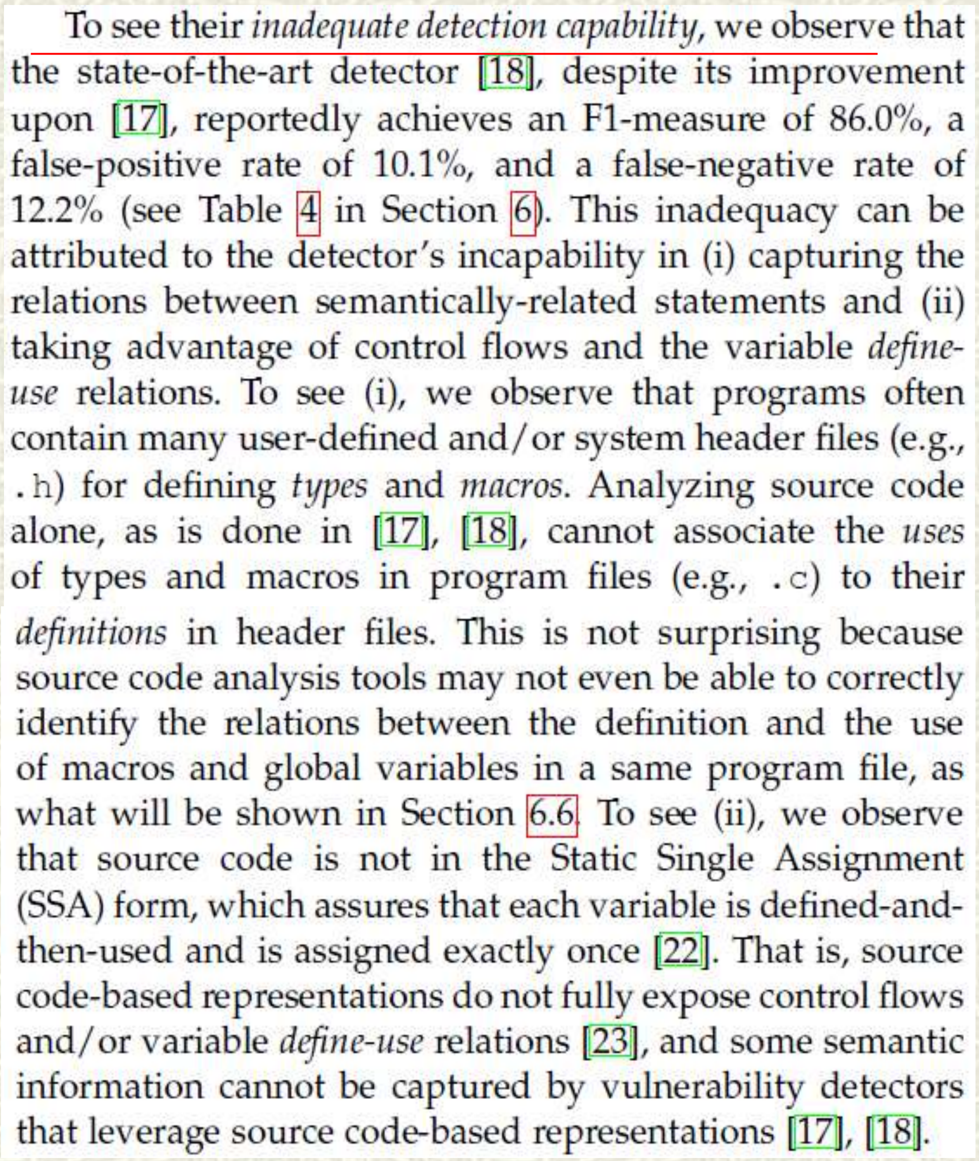


The recent development in *machine learning-based* vulnerability detection is to use deep learning [17], [18] and operate at a finer, or "program slice", level. These detectors also can relieve the problem of *manual-feature definition*, which has received further attention recently [19], [20], [21]. However, these detectors still offer *inadequate detection capability* and *inadequate locating precision*.

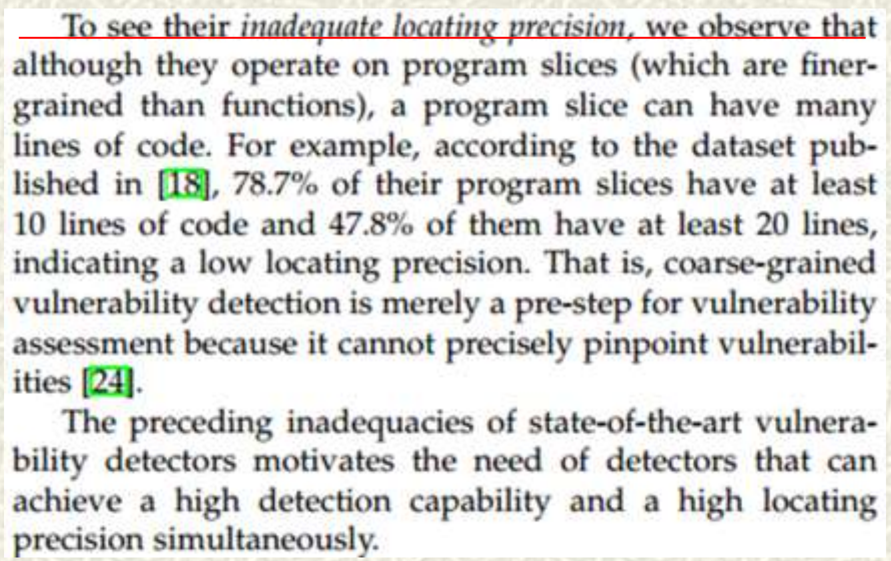
概述现有方法引出问题
检测能力不足
定位精度不够

英文引言示例

VulDeeLocator: A Deep Learning-based Fine-grained Vulnerability Detector



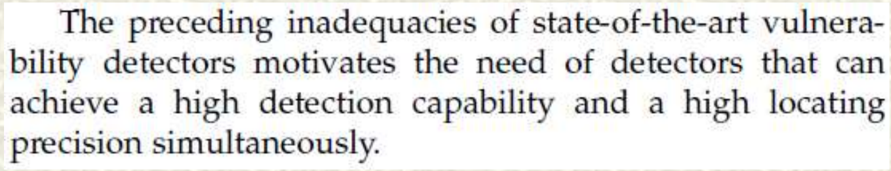
To see their *inadequate detection capability*, we observe that the state-of-the-art detector [18], despite its improvement upon [17], reportedly achieves an F1-measure of 86.0%, a false-positive rate of 10.1%, and a false-negative rate of 12.2% (see Table 4 in Section 6). This inadequacy can be attributed to the detector's incapability in (i) capturing the relations between semantically-related statements and (ii) taking advantage of control flows and the variable *define-use* relations. To see (i), we observe that programs often contain many user-defined and/or system header files (e.g., `.h`) for defining *types* and *macros*. Analyzing source code alone, as is done in [17], [18], cannot associate the *uses* of types and macros in program files (e.g., `.c`) to their *definitions* in header files. This is not surprising because source code analysis tools may not even be able to correctly identify the relations between the definition and the use of macros and global variables in a same program file, as what will be shown in Section 6.6. To see (ii), we observe that source code is not in the Static Single Assignment (SSA) form, which assures that each variable is defined-and-then-used and is assigned exactly once [22]. That is, source code-based representations do not fully expose control flows and/or variable *define-use* relations [23], and some semantic information cannot be captured by vulnerability detectors that leverage source code-based representations [17], [18].



To see their *inadequate locating precision*, we observe that although they operate on program slices (which are finer-grained than functions), a program slice can have many lines of code. For example, according to the dataset published in [18], 78.7% of their program slices have at least 10 lines of code and 47.8% of them have at least 20 lines, indicating a low locating precision. That is, coarse-grained vulnerability detection is merely a pre-step for vulnerability assessment because it cannot precisely pinpoint vulnerabilities [24].

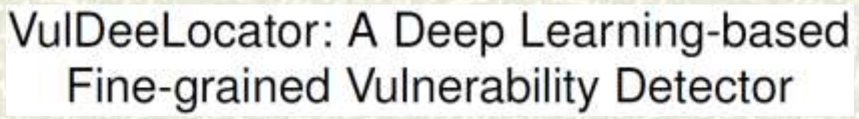
The preceding inadequacies of state-of-the-art vulnerability detectors motivates the need of detectors that can achieve a high detection capability and a high locating precision simultaneously.

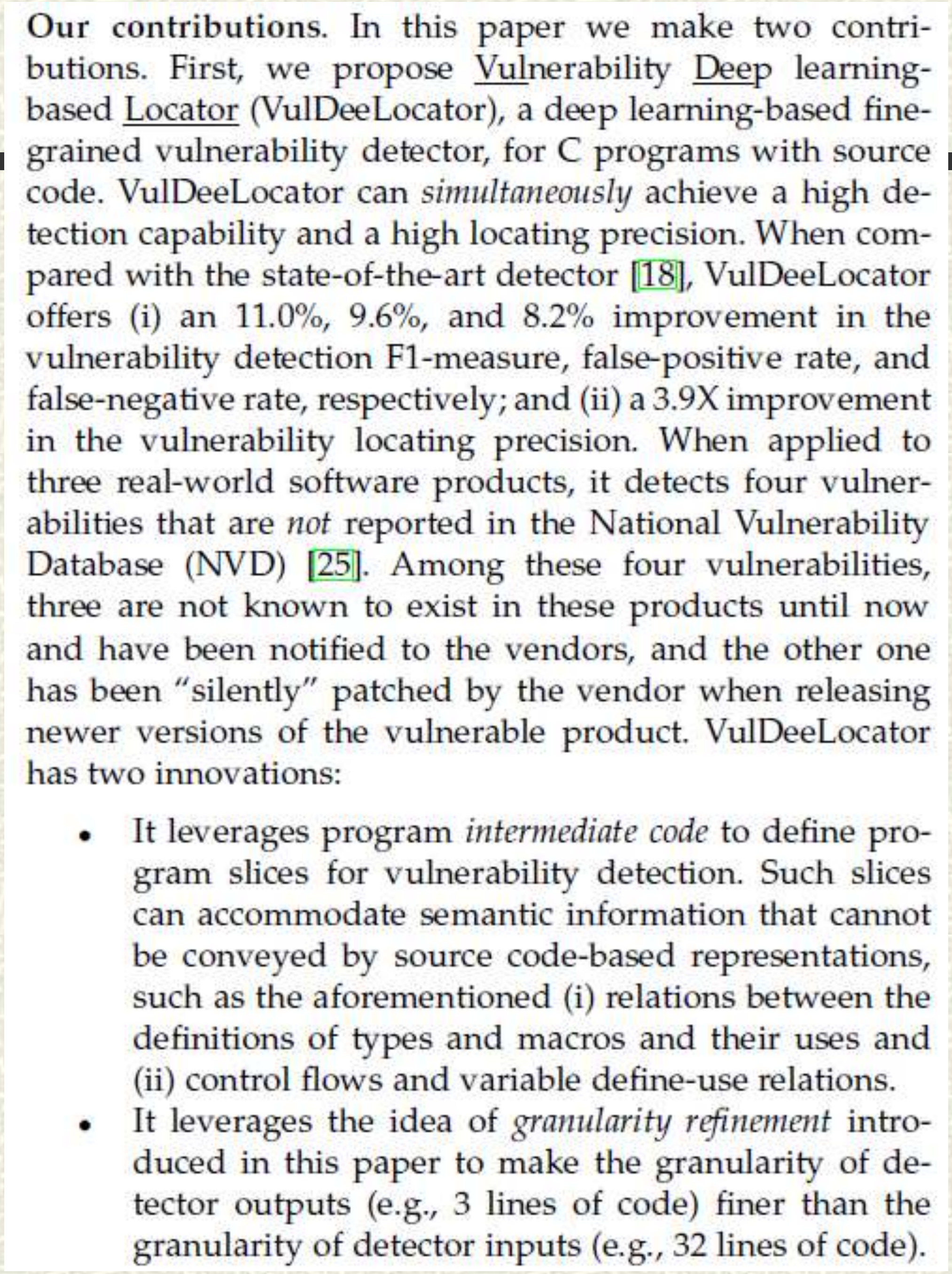
从宏观到具体，解释问题，为引出解决问题的方法做铺垫



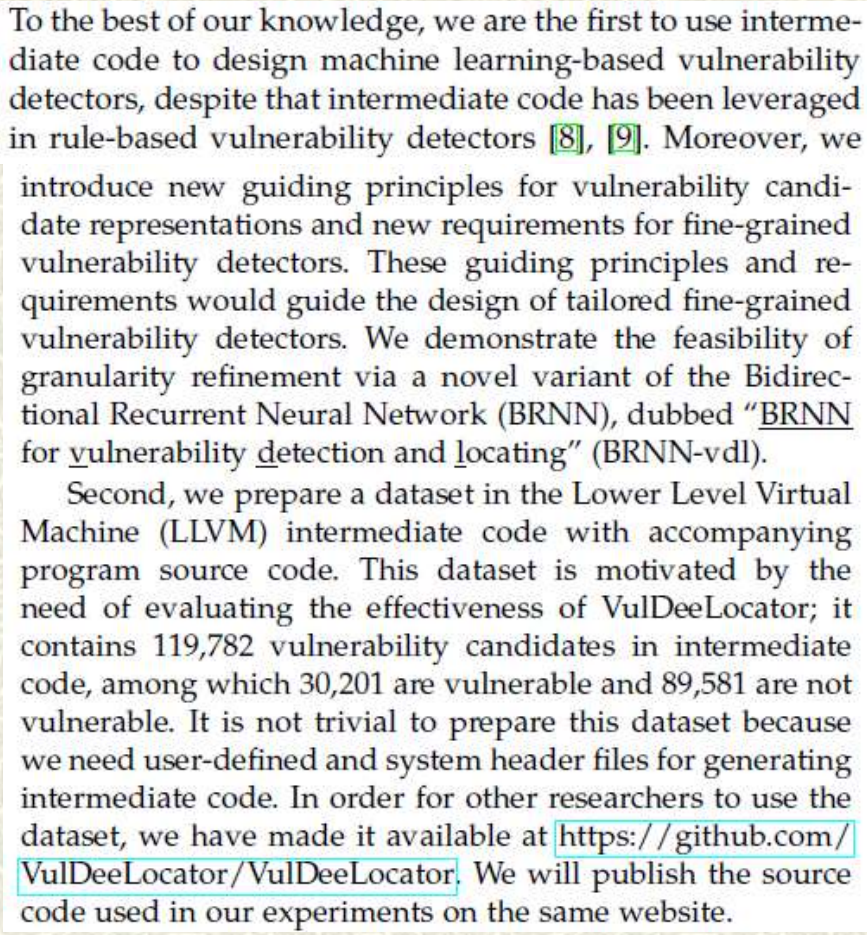
The preceding inadequacies of state-of-the-art vulnerability detectors motivates the need of detectors that can achieve a high detection capability and a high locating precision simultaneously.

问题分析引出研究目的


VulDeeLocator: A Deep Learning-based Fine-grained Vulnerability Detector


Our contributions. In this paper we make two contributions. First, we propose Vulnerability Deep learning-based Locator (VulDeeLocator), a deep learning-based fine-grained vulnerability detector, for C programs with source code. VulDeeLocator can *simultaneously* achieve a high detection capability and a high locating precision. When compared with the state-of-the-art detector [18], VulDeeLocator offers (i) an 11.0%, 9.6%, and 8.2% improvement in the vulnerability detection F1-measure, false-positive rate, and false-negative rate, respectively; and (ii) a 3.9X improvement in the vulnerability locating precision. When applied to three real-world software products, it detects four vulnerabilities that are *not* reported in the National Vulnerability Database (NVD) [25]. Among these four vulnerabilities, three are not known to exist in these products until now and have been notified to the vendors, and the other one has been "silently" patched by the vendor when releasing newer versions of the vulnerable product. VulDeeLocator has two innovations:

- It leverages program *intermediate code* to define program slices for vulnerability detection. Such slices can accommodate semantic information that cannot be conveyed by source code-based representations, such as the aforementioned (i) relations between the definitions of types and macros and their uses and (ii) control flows and variable define-use relations.
- It leverages the idea of *granularity refinement* introduced in this paper to make the granularity of detector outputs (e.g., 3 lines of code) finer than the granularity of detector inputs (e.g., 32 lines of code).

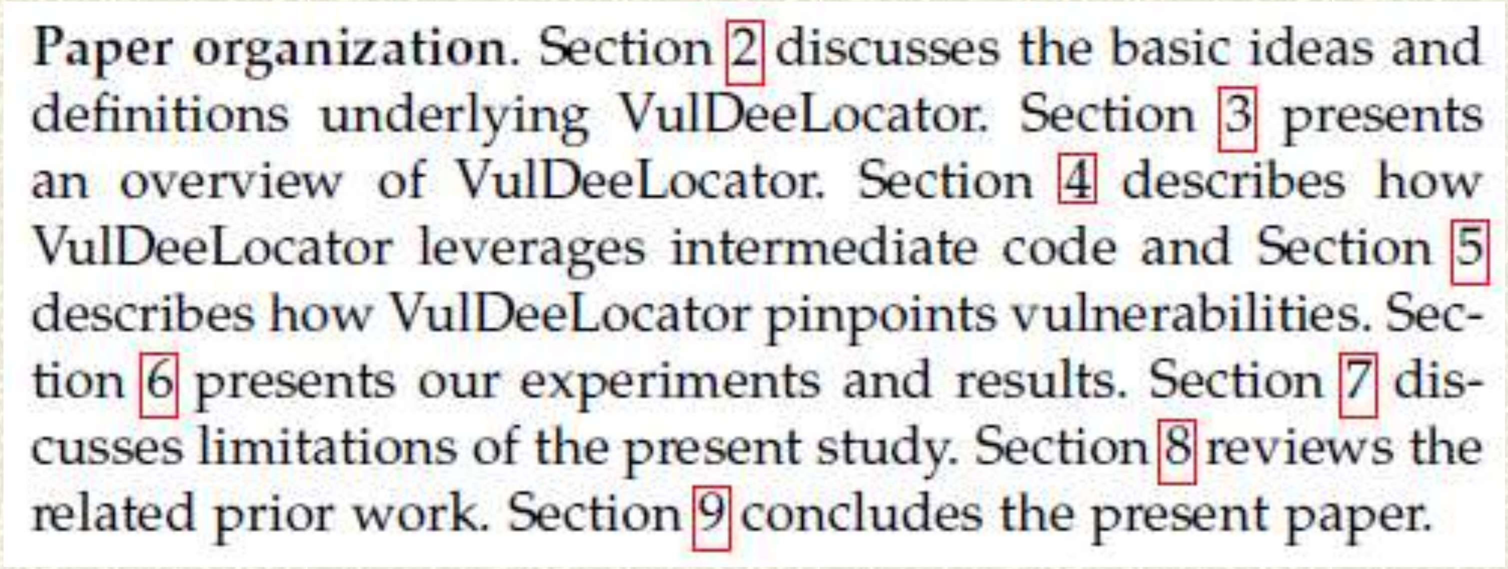

To the best of our knowledge, we are the first to use intermediate code to design machine learning-based vulnerability detectors, despite that intermediate code has been leveraged in rule-based vulnerability detectors [8], [9]. Moreover, we introduce new guiding principles for vulnerability candidate representations and new requirements for fine-grained vulnerability detectors. These guiding principles and requirements would guide the design of tailored fine-grained vulnerability detectors. We demonstrate the feasibility of granularity refinement via a novel variant of the Bidirectional Recurrent Neural Network (BRNN), dubbed "BRNN for vulnerability detection and locating" (BRNN-vdl).

Second, we prepare a dataset in the Lower Level Virtual Machine (LLVM) intermediate code with accompanying program source code. This dataset is motivated by the need of evaluating the effectiveness of VulDeeLocator; it contains 119,782 vulnerability candidates in intermediate code, among which 30,201 are vulnerable and 89,581 are not vulnerable. It is not trivial to prepare this dataset because we need user-defined and system header files for generating intermediate code. In order for other researchers to use the dataset, we have made it available at https://github.com/VulDeeLocator/VulDeeLocator. We will publish the source code used in our experiments on the same website.

在给出研究目的后列出主要贡献，并强调新在哪里

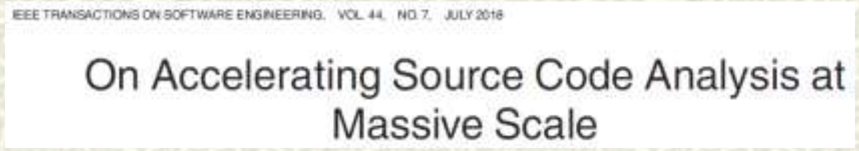
英文引言示例

VulDeeLocator: A Deep Learning-based Fine-grained Vulnerability Detector

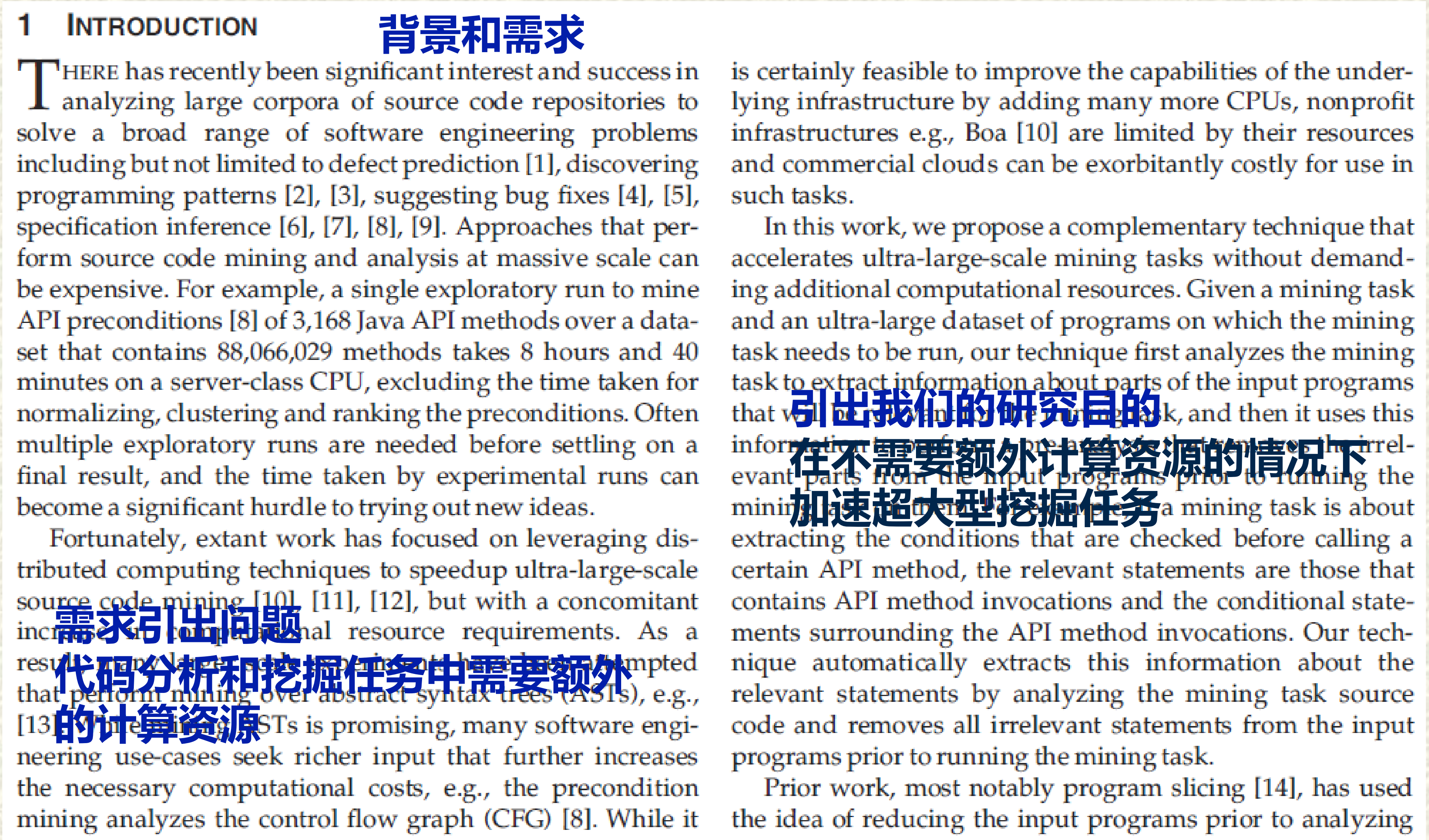

Paper organization. Section 2 discusses the basic ideas and definitions underlying VulDeeLocator. Section 3 presents an overview of VulDeeLocator. Section 4 describes how VulDeeLocator leverages intermediate code and Section 5 describes how VulDeeLocator pinpoints vulnerabilities. Section 6 presents our experiments and results. Section 7 discusses limitations of the present study. Section 8 reviews the related prior work. Section 9 concludes the present paper.

引言最后是组织结构

英文引言示例


IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 44, NO. 7, JULY 2018

On Accelerating Source Code Analysis at Massive Scale


1 Introduction

背景和需求

THERE has recently been significant interest and success in analyzing large corpora of source code repositories to solve a broad range of software engineering problems including but not limited to defect prediction [1], discovering programming patterns [2], [3], suggesting bug fixes [4], [5], specification inference [6], [7], [8], [9]. Approaches that perform source code mining and analysis at massive scale can be expensive. For example, a single exploratory run to mine API preconditions [8] of 3,168 Java API methods over a dataset that contains 88,066,029 methods takes 8 hours and 40 minutes on a server-class CPU, excluding the time taken for normalizing, clustering and ranking the preconditions. Often multiple exploratory runs are needed before settling on a final result, and the time taken by experimental runs can become a significant hurdle to trying out new ideas.

Fortunately, extant work has focused on leveraging distributed computing techniques to speedup ultra-large-scale source code mining [10], [11], [12], but with a concomitant increase in computational resource requirements. As a resu[illegible]mpted that perform mining over abstract syntax trees (ASTs), e.g., [13]. [illegible]STs is promising, many software engineering use-cases seek richer input that further increases the necessary computational costs, e.g., the precondition mining analyzes the control flow graph (CFG) [8]. While it is certainly feasible to improve the capabilities of the underlying infrastructure by adding many more CPUs, nonprofit infrastructures e.g., Boa [10] are limited by their resources and commercial clouds can be exorbitantly costly for use in such tasks.

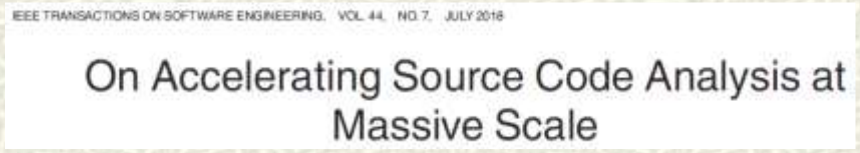
需求引出问题
代码分析和挖掘任务中需要额外的计算资源

In this work, we propose a complementary technique that accelerates ultra-large-scale mining tasks without demanding additional computational resources. Given a mining task and an ultra-large dataset of programs on which the mining task needs to be run, our technique first analyzes the mining task to extract information about parts of the input programs that w[illegible]k, and then it uses this infor[illegible]irrelevant parts from the input programs prior to running the minin[illegible]a mining task is about extracting the conditions that are checked before calling a certain API method, the relevant statements are those that contains API method invocations and the conditional statements surrounding the API method invocations. Our technique automatically extracts this information about the relevant statements by analyzing the mining task source code and removes all irrelevant statements from the input programs prior to running the mining task.

引出我们的研究目的
在不需要额外计算资源的情况下
加速超大型挖掘任务

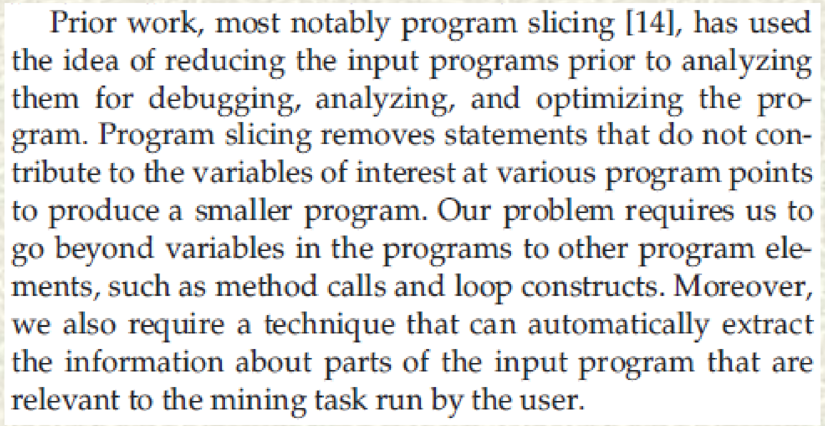
Prior work, most notably program slicing [14], has used the idea of reducing the input programs prior to analyzing

英文引言示例

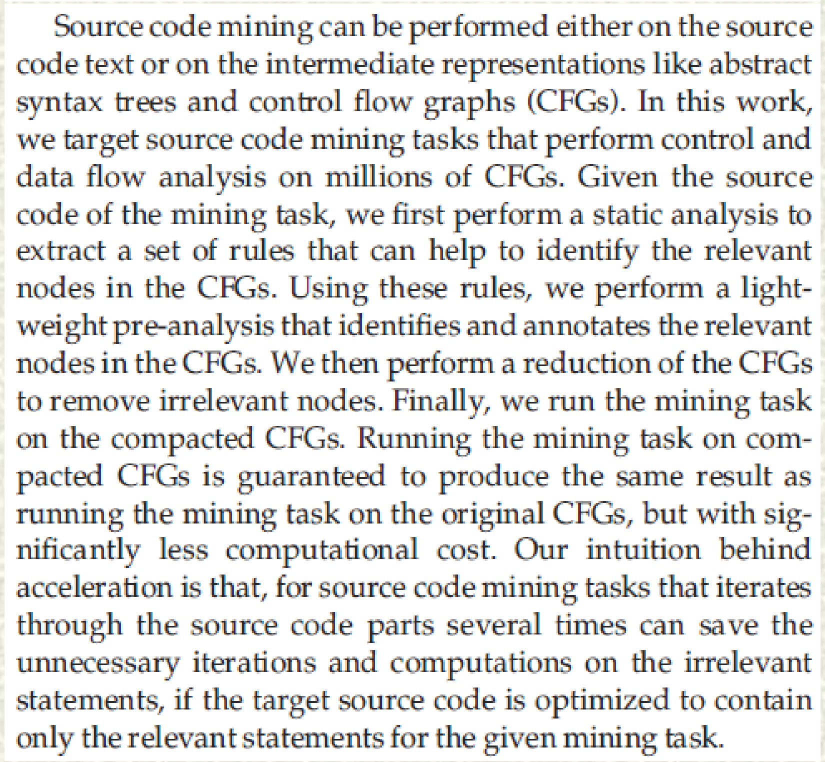


IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 44, NO. 7, JULY 2018

On Accelerating Source Code Analysis at Massive Scale



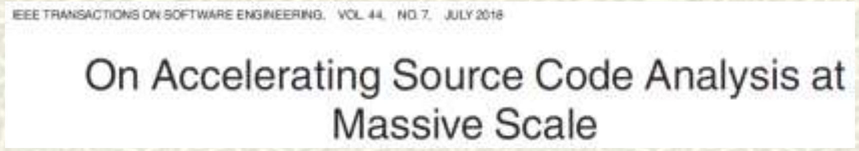
Prior work, most notably program slicing [14], has used the idea of reducing the input programs prior to analyzing them for debugging, analyzing, and optimizing the program. Program slicing removes statements that do not contribute to the variables of interest at various program points to produce a smaller program. Our problem requires us to go beyond variables in the programs to other program elements, such as method calls and loop constructs. Moreover, we also require a technique that can automatically extract the information about parts of the input program that are relevant to the mining task run by the user.



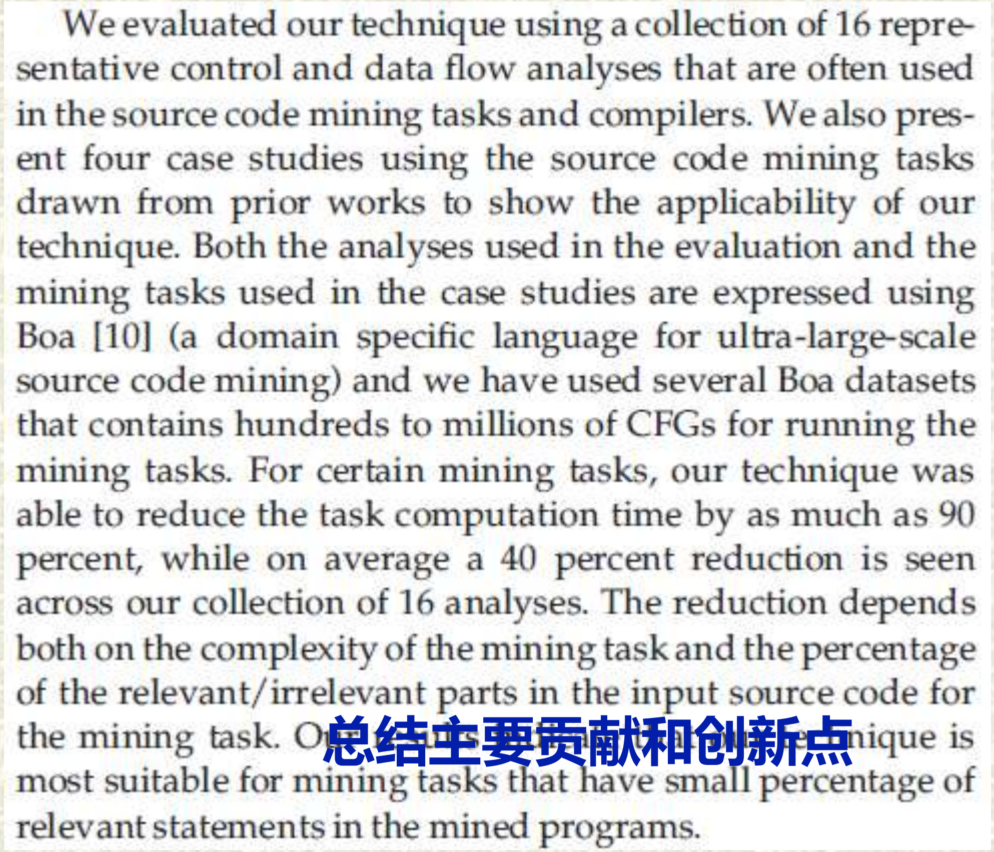
Source code mining can be performed either on the source code text or on the intermediate representations like abstract syntax trees and control flow graphs (CFGs). In this work, we target source code mining tasks that perform control and data flow analysis on millions of CFGs. Given the source code of the mining task, we first perform a static analysis to extract a set of rules that can help to identify the relevant nodes in the CFGs. Using these rules, we perform a lightweight pre-analysis that identifies and annotates the relevant nodes in the CFGs. We then perform a reduction of the CFGs to remove irrelevant nodes. Finally, we run the mining task on the compacted CFGs. Running the mining task on compacted CFGs is guaranteed to produce the same result as running the mining task on the original CFGs, but with significantly less computational cost. Our intuition behind acceleration is that, for source code mining tasks that iterates through the source code parts several times can save the unnecessary iterations and computations on the irrelevant statements, if the target source code is optimized to contain only the relevant statements for the given mining task.

概述之前的工作
解释我们思路的不同之处

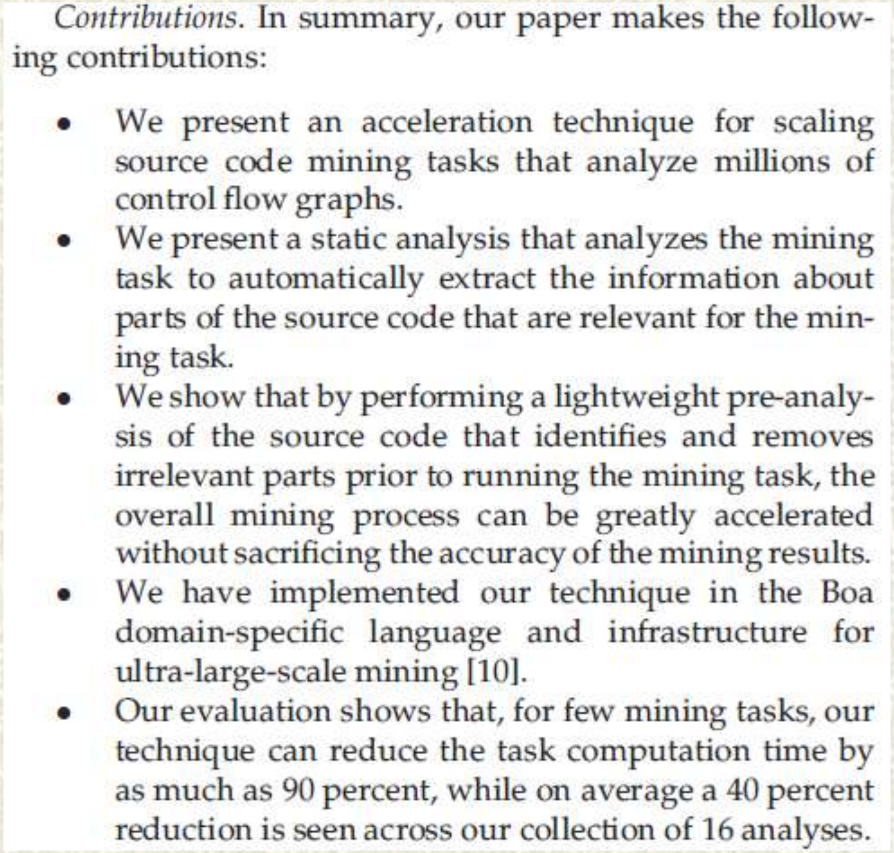
英文引言示例


IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 44, NO. 7, JULY 2018

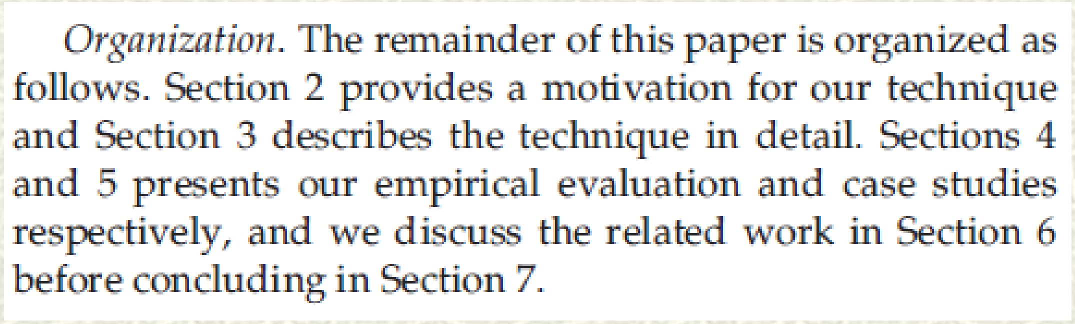
On Accelerating Source Code Analysis at Massive Scale


We evaluated our technique using a collection of 16 representative control and data flow analyses that are often used in the source code mining tasks and compilers. We also present four case studies using the source code mining tasks drawn from prior works to show the applicability of our technique. Both the analyses used in the evaluation and the mining tasks used in the case studies are expressed using Boa [10] (a domain specific language for ultra-large-scale source code mining) and we have used several Boa datasets that contains hundreds to millions of CFGs for running the mining tasks. For certain mining tasks, our technique was able to reduce the task computation time by as much as 90 percent, while on average a 40 percent reduction is seen across our collection of 16 analyses. The reduction depends both on the complexity of the mining task and the percentage of the relevant/irrelevant parts in the input source code for the mining task. Our results show that our technique is most suitable for mining tasks that have small percentage of relevant statements in the mined programs.

总结主要贡献和创新点


Contributions. In summary, our paper makes the following contributions:

- We present an acceleration technique for scaling source code mining tasks that analyze millions of control flow graphs.
- We present a static analysis that analyzes the mining task to automatically extract the information about parts of the source code that are relevant for the mining task.
- We show that by performing a lightweight pre-analysis of the source code that identifies and removes irrelevant parts prior to running the mining task, the overall mining process can be greatly accelerated without sacrificing the accuracy of the mining results.
- We have implemented our technique in the Boa domain-specific language and infrastructure for ultra-large-scale mining [10].
- Our evaluation shows that, for few mining tasks, our technique can reduce the task computation time by as much as 90 percent, while on average a 40 percent reduction is seen across our collection of 16 analyses.


Organization. The remainder of this paper is organized as follows. Section 2 provides a motivation for our technique and Section 3 describes the technique in detail. Sections 4 and 5 presents our empirical evaluation and case studies respectively, and we discuss the related work in Section 6 before concluding in Section 7.

引言最后是组织结构

中、英文论文引言在写作上的差异

- 篇幅差异
  - 英文期刊论文的“引言”一般较长，“故事情节”更丰富
    - 对研究概况、问题和创新点的阐述比较详实
  - 中文期刊论文的“引言”一般较短
    - 主要贡献和创新点，也可放在结论中

Thank you for your attention !

SuXiaoHong

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY