WASEF：Web 加速解决方案评估框架

MOUMENA CHAQFEH,NYUAD, 阿联酋RASHID TAHIR,University of Illinois Urbana-Champaign, 美国AYAZ REHMAN,LUMS, 巴基斯坦JESUTOFUNMI KUPOLUYI, NYUAD, 阿联酋SAAD ULLAH,LUMS, 巴基斯坦RUSSELL COKE, NYUAD, 阿联酋MUHAMMAD JUNAID,LUMS, 巴基斯坦MUHAMMAD ARHAM,LUMS, 巴基斯坦MARC WIGGERMAN,Vrije Universiteit Amsterdam, 荷兰ABIJITH RADHAKRISHNAN,Vrije Universiteit Amsterdam, 荷兰IVANO MALAVOLTA,Vrije Universiteit Amsterdam, 荷兰FAREED ZAFFAR,LUMS, 巴基斯坦YASIR ZAKI, NYUAD, 阿联酋

近年来，万维网变得日益复杂。这种复杂性严重影响了发展中地区的用户，原因是蜂窝数据连接速度缓慢，且他们使用的是低端智能手机设备。现有的简化网络解决方案通常使用多种不同的指标和设置进行评估，这使得难以相互比较这些解决方案。因此，要针对特定情境和使用场景选择合适的解决方案并不容易。本文介绍了 Wasef，这是一个框架，采用全面的时序、节省效果和质量指标集，以可复现的方式并在真实设置下评估和比较不同的网页复杂度解决方案。该框架整合了一系列现有的先进解决方案，并便于日后加入新方案。Wasef 首先通过爬取网页的首页和内页，创建网页缓存。随后，将缓存中的每个页面交由网页复杂性解决方案处理，生成优化版本。最后，在统一环境和指标下，以一致的方式评估每个优化版本。我们展示了该框架如何在真实条件下比较不同网页复杂度解决方案的性能特征。我们还通过对比发达国家的全球排名前 100 的网页与发展中国家中排名最低的 50 个国家的前 100 个网页进行评估，展示了发展中地区用户访问网页的可访问性可以得到显著提升。结果表明，这些网页的复杂性存在显著差异，并显示出我们的框架有望改善这些国家的网络无障碍。

1 引言

在过去十年中，万维网（WWW）的技术生态系统迅速发展，通过美观且高度交互型的页面，带来了更丰富、更吸引人的用户体验。然而，这些进步也导致页面变得显著更大、更复杂，页面大小中位数在过去fi五年中增长了 80% 以上 fi [39]。尽管ff智能手机设备价格日益亲民， [3——8, 11] 推动了移动网页浏览的增长（据报道，近期全球网络访问中有 68.1% 来自移动设备）， [13], 但网页复杂性正因网络连接缓慢以及这些国家普遍使用的低端硬件和价格低廉的手机设备，导致发展中国家的网页浏览体验不佳ff。 [28]在fi非洲、南美洲和印度， [21] 由于近期流fl量需求增加， [30]据报告，网页性能显著下降。

为改善糟糕的网页浏览体验，人们提出了多种有前景的解决方案，以应对网络日益增加的复杂性 [16, 23, 26, 31, 39, 45, 47, 48, 56, 62–64]。然而，每种解决方案的评估所采用的

页面集和指标各不相同，这使得解决方案之间难以相互比较，也妨碍开发人员根据自身偏好以及预期页面访客的需求选择合适的解决方案。此外，尽管这些解决方案有诸多益处，但由于缺乏广泛的使用途径，也缺少对其有效性和适用性的统一评估，因此难以对它们进行相互比较。

鉴于这些挑战，我们提出了一个框架，用于在受控实验环境中原型设计不同的评估场景，以便评估、比较并了解网页复杂性解决方案的优势。该环境能够生成可复现结果。我们的框架结合时序、节能和质量指标，并考虑多种场景，以实现全面评估。该框架采用一组现有解决方案，提供可扩展的实现，并支持轻松集成其他解决方案。框架会对不同类型的页面（包括着陆页和内页）进行网页爬取并缓存，以提供评估基线。对于每个网页，它会分别运行这些解决方案，为每种解决方案生成特定版本并缓存，以便进行比较分析。我们展示了如何在现实条件下使用该框架评估多种公开可用的最先进的网页复杂度解决方案的性能特征。我们分析了这些解决方案在一个包含 1000 个页面的数据集上的不同优势。我们还发现，发展中国家用户访问的页面往往比发达国家的热门页面更复杂；这一观察启发了一个实用的使用场景，即利用该框架改善这些国家的网络无障碍。总之，我们作出以下贡献：

- 提供一个评估框架，综合考虑不同的网页复杂度解决方案、网络状况和性能指标。
- 维护一个具有代表性的评估数据集，包含首页和内部页面。
- 分析一系列公开可用的网页复杂度解决方案的优势，展示其性能提升对首页和内部页面的结构与功能产生的影响。
- 分析发展中国家的 100 个热门网页，并与全球排名前 100 的热门网页进行比较，展示两者在复杂性方面的差异，以及我们的框架所能带来的潜在优势。

2 动机

现有文献提出了多种应对网页日益增长的复杂性的方案，包括图像压缩 [16, 39], 页面加载重构 [47, 56, 63],以及移除非必要元素的 JavaScript 优化 [26] 或消除未使用的函数 [40, 45, 50]。这些方案分别通过修改原始网页中的一组网页组件（如图像）或流程（如页面加载），来生成性能更优的版本，并根据具体目标采用不同指标ff进行评估fi。目标包括但不限于：加快页面加载、节省内存、降低能耗以及适应用户的数据套餐。

然而，未能展示网页复杂性解决方案对某项关键测量指标的影响，可能会导致人们对其优点产生错误判断，或误用该解决方案。例如，节省内存的解决方案 [45]可能无法实现时间性能改进，因此不能提供给存在连接问题的用户使用，因为这类用户非常依赖页面加速。此外，用于加快页面加载的任何优化，都可能影响页面内容或交互功能 [26]。因此，只有通过全面评估，才能清晰了解其优点和缺点，从而根据用户需求和设置使用合适的解决方案。

传统指标可分为三ff类： 时间指标、能耗和页面质量指标。关于这些指标的另一种分类方式见fi。在表 [20]中，我们列出了用于评估最先进的网页复杂度解决方案的指标。如表中所示，这些解决方案大多选择测量时间指标

或能耗（内存、CPU 和电池），而忽视了从内容完整性和功能保留角度进行的质量评估。尽管如此，仍有少数方法 [26, 39, 45] 采用小规模用户研究来衡量生成页面的质量。这可以用开展大规模用户研究所面临的挑战来解释，主要是由于 fi资金和/或后勤限制。下文将讨论比较网页复杂性解决方案以进行性能分析时面临的主要挑战。

访问受限： 如今大多数网页复杂度解决方案并未公开。随着时间推移，即便是前景看好的解决方案，也可能因复现和应用困难而被社区弃用。这项工作的动机之一，是营造一种鼓励研究人员提高其解决方案可访问性的氛围，方法是将这些方案整合到统一的测试框架中，以便进行比较、一致且标准化的评估。

可复现性：即使使用相同的页面列表，由于万维网的复杂性迅速增长，复现对给定网页复杂度解决方案的评估结果仍然具有挑战性。我们提出的框架旨在“冻结”一个大型网页数据集，以便对不同解决方案进行一致且可复现的评估。我们可以持续爬取这些页面的新实例，确保其持续反映网络现状；同时存储每个页面的冻结版本，并报告数据收集日期，以便日后与相同页面进行比较。

工具的局限性和测量指标的快速变化 不同的网页复杂度解决方案使用不同的工具进行评估。因此，若不进行进一步实验，可能无法对这些解决方案开展比较分析。此外，现有工具均未涵盖上述三类测量指标，若不混合使用各种独立工具，就无法全面了解某个解决方案的优缺点。现有工具会定期推出新版本，以提供更完善的测量指标并适应万维网的快速演变，这一事实进一步加剧了这些挑战。例如，Google Lighthouse 6 用更新且更具代表性的用户感知时间指标，替换了旧版本中的一组指标。此外，它还引入了新的评分算法，对先前版本所采用的测量指标计算方式做出了多项调整。在同一组页面上比较版本 5 和版本 6 后发现，96% 的页面最终得到了不同的分数 [17]。这意味着，使用 Lighthouse 5 评估的解决方案，若不经过进一步评估，就无法与使用 Lighthouse 6 评估的解决方案进行比较。

评估对页面内容和功能的影响： 网页复杂性解决方案通常会从现有页面生成简化版本，而由于对资源和/或流程进行优化，原始内容和/或功能可能会有所牺牲。因此，对生成的页面进行结构/功能评估对于确保页面质量至关重要。然而，由于开展大规模用户研究面临挑战，且缺乏其他评估场景，这类评估要么被忽略，要么通过小规模用户研究进行。

3 相关工作

3.1 网页性能测量工具

有许多工具可用于评估网页性能。WebPageTest [22] 是一款广泛使用的开源网络性能测量与分析工具，提供面向社区的在线版本，并由公司和个人赞助，以便在全球范围内提供测试代理基础设施。它允许用户

表1。最先进的网络复杂度解决方案的评估指标（PLT 为页面加载时间）。

解决方案	时序	保存	质量
WProf [62]	PLT	-	-
Flywheel [16]	PLT, 首字节时间first, 首次绘制时间first	页面大小	-
Shandian [63]	PLT	页面大小	-
Polaris [47]	PLT	-	-
Vroom [56]	页面加载时间，首屏时间，速度指数	-	-
Prophecy [48]	页面加载时间，速度指数就绪指数,	带宽, 能耗	-
SpeedReader [31]	PLT	数据大小, 内存	-
JSCleaner [26]	页面加载时间、DOM 完成时间	页面大小	用户研究
Web Medic [45]	-	内存	用户研究
BrowseLite [39]	速度指数	带宽, 页面大小	用户研究, 结构

从不同地点评估网页、验证更改并审核性能。我们认为，WebPageTest 的功能使其成为集成和评估网页复杂度解决方案、并促进这些方案在发展中国家适用性的理想环境。在本研究中，我们扩展了 WebPageTest，以实现这一目的。除 WebPageTest 外，还有一系列公开可用的工具可用于评估页面的不同方面。例如，Lighthouse ff是一款用于网页性能测量的开源工具，它会针对指定页面依次执行一系列审核，然后生成可用于改进页面的报告。同样，也可以使用 SEMrush [12] 工具报告一系列常见的网络性能问题。其他商业工具则可跨不同 [2] 浏览器、设备和地点测量网络性能，例如 ff [14] 和 [10]。尽管上述工具提供了高效的网络性能评估环境，但它们主要关注对给定网页单一版本的时序测量，而复杂性解决方案可能提供的不同ffi版本仍不在其当前评估范围内。通常会通过小规模用户研究，将这些版本与给定网页的原始版本进行比较（见第 2节）。为应对用户研究在规模扩展和自动化方面的挑战，Eyeorg 的作者提出了一种众包网页质量测量系统，并在受控实验条件下评估页面结构变化对真实用户的影响。相比之下，我们的框架依靠计算机视觉，以更大规模地评估网页内容结构。此外，利用计算机视觉还使我们能够评估网页功能，而 Eyeorg 缺少这一特性。评估网页的能耗、CPU 和内存使用量，对于评估移动设备上的网页浏览至关重要。Android Runner ff是一款可在 Android 设备上自动执行实验并测量这些指标的工具；我们的框架集成了该工具以实现这一目的。其他现有工具要么仅针对单项测量（例如能耗）而设计，要么是为特定实验临时开发的ff。 [61] [43]

3.2 网络测量指标研究

现有的网页研究主要侧重于深入了解网页跟踪 [29, 38, 41, 57], 广告拦截 [36,44, 58], 以及安全和隐私 [37, 51, 52, 60], 的现状，较少关注性能、可访问性和用户体验。

例如，文献 [57] 和文献 [29], 通过约 1500 个页面 [38], 以及文献 [41]在二十年间（从 1996年 到 2016年）对数百万个页面的网页跟踪进行了分析。在 [44], 中，研究了广告拦截器的影响；而 [36], 则测量了反广告拦截情况，并在 [58]中借助分析框架展望了广告拦截的未来。此外， [42]对一百万个网站上的第三方请求进行了分析。在网页分析的性能研究方面，近期一项研究评估了 150,000 个页面（包括桌面端和移动端）的性能 [59], ，发现所分析的页面中有 82% 存在显著的fi性能问题，其中 44% 至少存在一个严重问题。这项研究表明，现代网页普遍存在性能问题，也促使人们需要应用解决方案，帮助使用低端移动设备访问网页的弱势用户。为了解不同ff因素对改善移动网页加载的影响，另一项近期研究 [46] 历时 4 年，对 8 款热门移动浏览器和 250,000 个页面进行了研究，发现网页日益复杂，而移动网页性能的提升主要归功于移动设备平台的改进（而非页面或浏览器/网络的改进）。据我们所知，我们对 1000 个网页的分析是 fi首个展示网页复杂性解决方案对不同ff性能指标影响的研究，旨在推动大规模应用这些解决方案。

4 WASEF 框架

Wasef 框架旨在通过为各种解决方案提供平台，并在 Android 手持式智能手机设备上进行测量，克服网页复杂度解决方案性能评估中的现有挑战。除了文献中对 Android 的广泛关注之外， [19], 我们选择 Android 智能手机，旨在评估在发展中国家广受欢迎的不同设备上的网页复杂度解决方案 [4–8]。本框架的其他核心设计考量包括：

- **自动化：**除初始配置和设置外，无需用户干预。
- **可扩展性：**只需少量集成工作，开发人员和研究人员即可将自己的网页复杂度解决方案集成到框架中，并通过用户界面使其可用。
- **可复现性：**在给定配置、数据集和智能手机设备的条件下，实验可以复现。
- **全面性：**同时提供时序和能耗测量指标，以评估每种解决方案在着陆页面和内部页面上的优势与不足。

- **评估的可扩展性：**快速、自动化地大规模评估网页复杂度解决方案所生成页面的结构和功能，从而提供一种替代方案，克服用户研究的挑战与局限。

4.1 架构

Wasef框架架构的高层概览如图所示1。Wasef包含三个主要阶段：处理（爬取网页、在每个页面上运行网页复杂度解决方案，并缓存这些解决方案生成的优化后页面版本）、评估（运行测量实验）以及结果收集（呈现并输出所开展实验的统计数据）。

在处理阶段，系统会爬取一组网页，并将其缓存到专用代理服务器中。为了提供具有代表性的网页集合，同时由于着陆页与内部页面在内容和结构方面存在显著差异（如 [18], 所述），因此同时考虑了着陆页和内部页面。这些页面代表将通过一组集成式网页复杂性解决方案创建的页面的原始版本。我们根据以下标准选择了以下网页复杂性解决方案：Lacuna [50],JSCleaner [26],SlimWeb [25],Muzeel [40],和 JSAnalyzer [24],：1）源代码的可用性；2）生成简化页面的潜力；

该页面源自原始版本；3）在手持移动电话设备上的适用性。通过在每个原始页面上运行这些解决方案，会生成一个简化页面，并将其缓存到代理服务器中。爬取的原始页面及其简化版本为在各种配置下进行评估提供了一个数据集。该评估包括三项主要的基于实验的测量：时序（页面加载以及一组用户感知的时序指标）、能耗（电池消耗、CPU 和内存使用量），以及相对于原始版本对简化页面进行的结构/功能评估。据我们所知，这三项主要测量涵盖了当今网络社区所考虑的全面评估指标，但这些指标尚未在统一框架中结合提供。

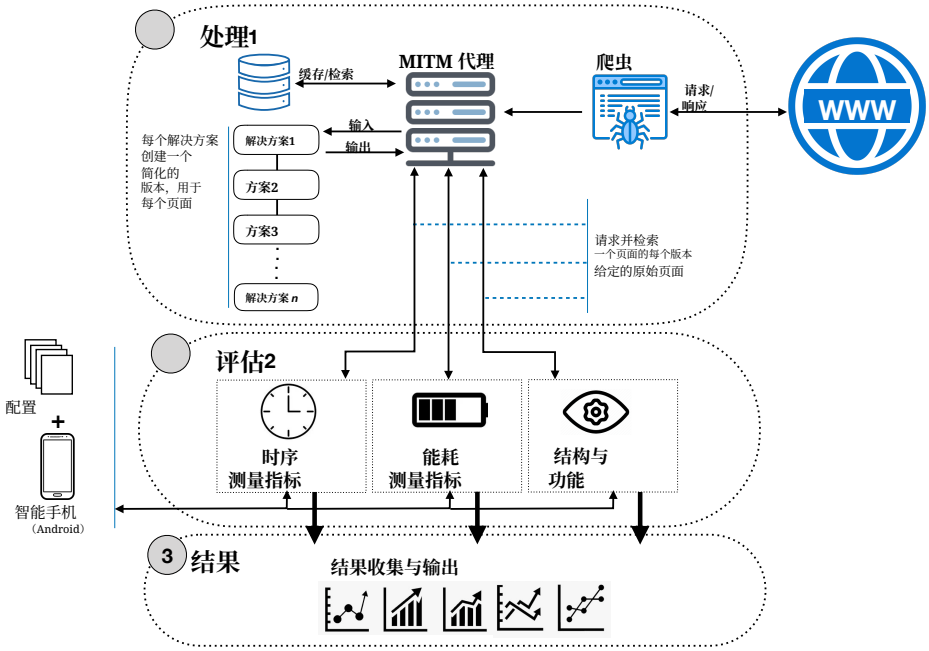


图1. Wasef框架架构

4.2 性能测量

我们的框架考虑了三组性能指标，以进行全面的比较分析：时序、能耗以及页面结构/功能质量指标。

4.2.1 时间测量。经典时序指标是页面加载时间（PLT），但它无法充分反映真实用户的体验。因此，人们提出了可交互时间 [49], 和速度指数 [34]等时间指标，以衡量页面加载过程中用户感知到的性能。我们的框架涵盖 WebPageTest 的所有时间指标 [22]，范围从 fi首次内容绘制到完整页面加载时间。除时间指标外，我们还报告带宽消耗测量结果，包括页面下载大小和网络请求数量。

4.2.2 能耗测量。这些测量指标包括电池、CPU 和内存消耗，并通过 Android 实用工具收集，具体如 [43]所述。

4.2.3 结构与功能评估。我们采用两项指标评估网页复杂度解决方案生成的页面相较于原始版本的结构和功能质量：1）结构相似性，即对简化后的页面进行评分，以评估其相较于对应原始版本的内容完整性，包括文本、图像和交互组件；2）功能相似度，即对简化后的页面进行评分，以评估其相较于原始页面的功能完整性。这项评估采用计算机视觉，作为大规模用户研究的替代方法：通过与对应原始页面的相似度评分来评估给定简化页面的结构；同时，通过模拟用户与简化页面和原始页面的交互，评估该页面的功能，并给出功能完整性评分。

4.3 实现

我们扩展了现有的 MITM 代理实现，并结合 Selenium [35] 和 Chrome 驱动程序，从网络上爬取网页并将其缓存到专用代理服务器中。为了将各项网页复杂度解决方案集成到我们的框架中，我们扩展了 WebPageTest [22] 工具，使用户能够通过该工具的用户界面选择这些解决方案。为了集成能耗测量，我们扩展了 AndroidRunner [43] 框架，该框架可在 Android 设备上自动执行实验，以测量电池消耗、CPU 和内存使用量。此外，我们集成了 Qlue [32, 33]，用于评估采用的复杂度解决方案所生成页面相较于其原始版本的结构相似性和功能相似度。网页开发人员和研究人员可以通过两种方式扩展 Wasef 框架。首先，集成自己的网页复杂度解决方案，以生成新的一组简化页面。其次，将自己的测量工具集成到框架中，利用 AndriodRunner 的插件处理程序 [43]所提供的扩展点实现这一功能。

5 评估

我们的评估旨在展示 Wasef 在创建统一环境以评估不同网页复杂度解决方案方面的优势，并从时序、功耗、页面结构和功能保留等角度对所采用的各项解决方案进行比较分析。我们的评估包含两项实验：1）对一组公开可用的网页复杂度解决方案进行比较分析，其中每种解决方案都用于生成一组由 1000 个原始网页构成的集合的简化版本；2）分析全球最受欢迎的 100 个页面与一组在发展中国家广受欢迎的 100 个页面的性能。典型的 Wasef 实验会使用一台服务器运行软件模块，按顺序执行一系列实验，以评估一组网页，包括原始页面以及由所采用的解决方案生成的简化版本。

5.1 实验设置

我们专门配置了一台配备 64 个核心和 1 TB 内存的高性能服务器，用于爬取并缓存一组包含 1000 个热门网页的页面，其中既有落地页，也有内部页面；这是 Hispar [18] 最近建议的做法。我们从 Hispar 列表 [55]中选取了 500 个页面，另从 Tranco [15, 53]中选取了 500 个页面（该列表收录了来自 Alexa、Cisco Umbrella 和 Majestic 的热门网页）。从每组中的第 fi个页面开始，我们 fi筛除了屏幕截图小于 1 MB 的页面，以排除轻量级网页。选定的 1000 个页面通过 Chrome浏览器加载，同时记录完整的

HTTP(S) 内容和标头，使用扩展版 MITM 代理 [27]。服务器还托管了 Wasef 软件模块，用于：1）在缓存网页上运行所采用的网页复杂度解决方案，并存储每种解决方案生成的输出版本；2）依次考量所有测量指标，对每个版本进行全面评估，首先是时间指标，其次是能耗，最后是结构/功能计算。对于给定的原始网页，除 JSAnalyzer 页面外，所有版本均自动生成；JSAnalyzer 页面由多名受聘的网页开发人员创建，再由服务器缓存。为评估这些页面与其原始版本之间的结构相似性，我们监控了屏幕截图的生成，以确保截图质量；滚动时反复出现的元素（例如 fl悬浮横幅）会被移除。

此外，为了为第二项实验创建数据集，我们从 Tranco 2021年全球热门网站排名前百万的网站中，爬取了全球热门页面排名前100的页面。我们还从同一列表中，爬取了全球发展中国家列表中排名最低的50个国家所对应的网站中排名前100的网站，并根据人类发展指数 [1] 进行排序。

为了在两项实验中模拟慢速蜂窝网络，我们将 Wasef 配置为模拟 3G网络，其带宽为1.6 Mbps，往返时间（RTT）为400毫秒。我们选用了 Xiamoi RedmiGo 作为具有代表性的低端 Android 智能手机设备（操作系统：Android 8.1 Oreo（Go版），CPU：四核1.4 GHz Cortex-A53，内存：1GB，电池：锂离子3000 mAh），该设备通过高速 WiFi 连接互联网（上传和下载带宽均为100 Mbps）。系统通过在手机上启动 Chrome浏览器，自动请求页面；该浏览器是最受欢迎的浏览器 [9]。

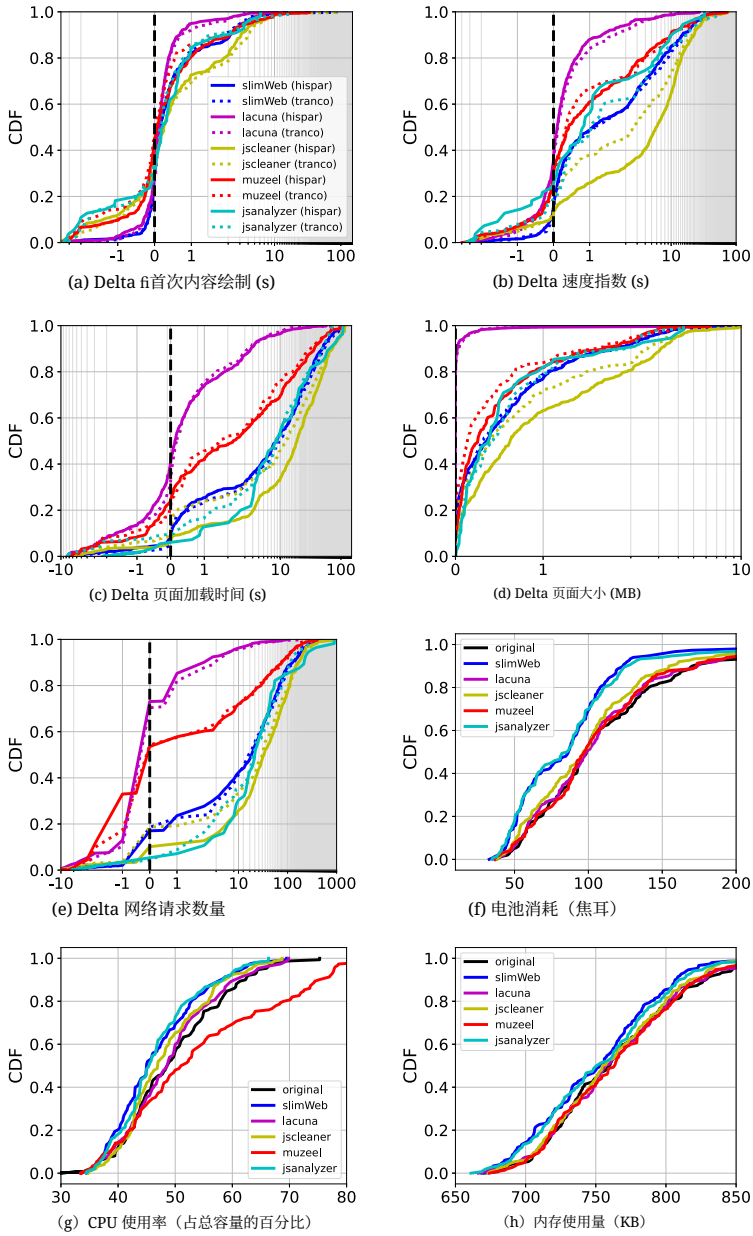
JSAnalyzer 页面生成： *JSAnalyer* 由 10 名用户进行评估。这些用户通过社交媒体群组获邀使用 JS Analyzer 生成简化页面，邀请方式包括介绍该工具并提供联系邮箱。收到的请求按照先到先得的原则接受。每位用户都会获邀参加在线培训课程，以分析一个网页。用户可以从一组可用的日期和时间中选择一个小时时段。在每次课程中，都会向用户介绍 JSAnalyzer 的用途，随后快速讲解其使用方法。用户可以安装 JSAnalyzer，以分析现有网页并生成新的简化页面。当用户成功评估单个页面并保存简化版本后，一次在线评估课程即告结束。每次课程所用的时间取决于待分析页面的复杂程度（5-30 分钟）。在线评估课程结束时，用户会被要求继续离线分析一组共 15 个页面，最终由 10 名用户完成对总计 150 个页面的分析。

5.2 网页复杂性解决方案分析

在本节中，我们对 fi种不同的最先进的网页复杂性解决方案——即那些公开发布了代码的方案——进行了相互比较。评估从三个不同的高层性能角度对这些解决方案进行比较，具体如下：

- 时间指标：突出显示页面元素的加载速度。
- 基准测试指标：突出显示页面使用的不同资源。
- 定性指标：突出显示页面与其原始对应页面在结构和功能上的相似度。

此外，还在两组不同的页面上对每种解决方案进行了评估：500 个来自 Tranco列表 的页面（代表全球访问量最高的落地页），以及 500 个来自 Hispar列表 的页面（代表全球访问量最高的内部页面）。



(a) Delta fi首次内容绘制 (s)

(b) Delta 速度指数 (s)

(c) Delta 页面加载时间 (s)

(d) Delta 页面大小 (MB)

(e) Delta 网络请求数量

(f) 电池消耗（焦耳）

（g）CPU 使用率（占总容量的百分比）

（h）内存使用量（KB）

图 2. Qu网页复杂度解决方案的定量比较

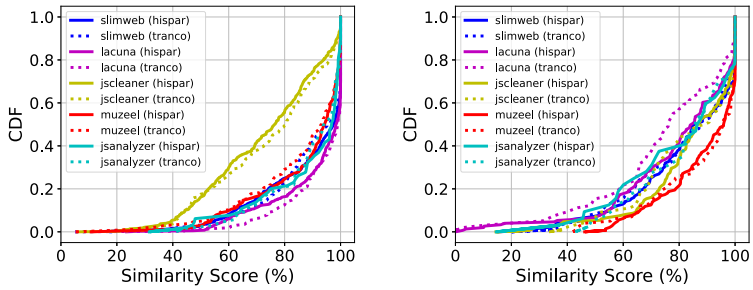
时间指标：图 2(a) 至 2(e) 展示了各页面版本（以生成该版本所用的解决方案命名）相对于对应原始页面的性能差值结果的累积分布函数。计算差值时，需从原始页面测得的指标值中减去特定页面版本测得的指标值。图 2(a) 展示了首次内容绘制指标的差值，这是一项时间指标，用于衡量浏览器多快能显示首批视觉元素。从图中可以看出，在不同解决方案中，JSCleaner（以黄色曲线表示）的差值最高。另一方面，Lacuna（以洋红色曲线表示）在所有解决方案中的差值提升最低。图中还显示，应用于落地页与内部页面时，各解决方案之间的差异微乎其微。接下来，图 2(b) 展示了速度指数指标的结果。速度指数是一项常用的时间指标，用于衡量页面加载期间内容以视觉方式显示的速度。从图中再次可以看出，与所有其他解决方案相比，JSCleaner 的速度指数差值高出许多；与原始页面相比，其速度指数中位数大约快了六秒。此外，JSCleaner 似乎对内部页面速度指数的提升效果优于落地页。图 2(c) 展示了所有解决方案的页面加载时间累积分布函数结果。结果表明，与 JSCleaner 相比，SlimWeb 和 JSAalyzer 的页面加载时间差值略低。另一方面，与其他所有解决方案相比，Lacuna 的页面加载时间差值提升非常有限。页面大小差值（图 2(d)）和网络请求数量差值（图 2(e)）也呈现出类似趋势。

基准测试指标：为研究网页复杂度解决方案对移动电话所用资源的影响，我们将三项不同的指标进行比较：ff“电池消耗”、“CPU 利用率”和“内存使用量”。图2(f) 至2(h)分别展示了所有版本及原始版本所报告测量指标的累积分布函数。对电池消耗结果的分析表明，这些解决方案基本分为两组：第一组包括ff和fi，即*SlimWeb*和*JSAnylzer*；第二组则包括其余解决方案。与其他解决方案相比，第一组 fi的电池消耗显著更低，这表明*SlimWeb*和*JSAnalyzer* 都通过阻止非必要JavaScript降低了电池消耗。对于另外两项基准测试指标 CPU 和内存，可以看出各解决方案之间没有明显差异，但ff除外：*Muzeel* 在约 60% 的页面上似乎比其他解决方案消耗更多 CPU。

定性指标： 任何网页复杂性解决方案的分析都不完整，除非评估该解决方案对页面视觉内容和功能的影响，即评估其与原始页面在内容和功能上的相似度。因此，Wasef 使用 Qlue 计算上述 fi网页复杂性解决方案生成的所有页面与原始页面之间的结构相似性和功能相似度，如图 3所示。

图 3(a) 展示了结构相似性的累积分布函数，可以看出 *JSCleaner* 与其他所有解决方案相比，其相似度得分最低，中位结构相似度得分约为 80%；而其他所有解决方案的相似度得分均保持在 95%以上。观察图 3(b) 中的功能相似度得分可以发现，除 *Muzeel* 外，所有解决方案的中位功能相似度得分均约为 85%。相比之下，*Muzeel* 相较于原始页面保留了超过 95%的功能相似度。

结语： 总而言之，为总结网页复杂度解决方案的比较结果，我们需要综合评估所有指标。尽管时间指标显示，*JSCleaner* 在 fi首次内容绘制、速度指数、页面加载时间、页面大小和网络请求数量方面均优于其他解决方案，但



(a) 页面结构评估　　(b) 页面功能评估

图 3. Qu化相似度比较

这些节省会对页面结构（见图 3(a)）和功能（见图 3(b)）造成影响。相比之下，*Lacuna* 在图 2(a) 至2(e)所示的各项指标中，性能提升最少，但其结构相似度得分最高（见图 3(a)）。另一方面，*Lacuna* 对页面功能的影响可以归因于页面加载后缺少用户交互性评估，因此将交互功能标记为未使用代码。这一局限已在*Muzeel* 中得到解决（该方案同样依赖死代码消除），其功能相似度有所提升（见图3(b)）。

上面的这两个示例（*JSCleaner* 和 *Lacuna*）清楚地表明，仅根据部分指标评估网络复杂度解决方案并不足以得出明确结论。这也凸显了使用 Wasef 这类框架的优势：它可以从不同角度进行全面分析，以研究此类解决方案的影响。

有趣的是，与 *SlimWeb* 相比，*JSAnalyzer* 显著缩短了页面加载时间，并减少了网络请求数量。在 CPU 使用率方面，*Muzeel* 的 CPU 利用率在所有解决方案中最高。还可以看出，与其他解决方案相比，*slimWeb*和 *JSAnalyzer* 等拦截型解决方案通常具有较低的电池消耗。最后，所有不同解决方案对内存消耗的影响都不显著。

5.3 网页复杂度：发展中地区与发达地区

本节使用 Wasef 框架评估发展中地区排名前 100 的页面与发达国家的全球排名前 100 的页面之间的差异。此项分析旨在了解发达国家页面与发展中地区页面在页面复杂性以及网页复杂性解决方案在这些页面上的潜在效果方面有何不同。我们爬取了 2021 年 Tranco 列表中的全球排名前 100 的页面，以及根据人类发展指数排序后，排名处于较低三分之一的发展中地区在 2021 年的前 100 个页面 [1]。我们选择 *SlimWeb* 作为要在这两组页面上评估的网页复杂性解决方案，以量化其对页面加载速度的提升效果。鉴于 *SlimWeb* 通过屏蔽非必要 JavaScript 来缩短页面加载时间，我们评估了它对浏览器整体 JavaScript 处理时间以及页面整体加载时间的影响。

[1]该列表取自以下来源中列出的 152 个发展中地区的完整名单https://www.worlddata.info/developing-countries.php。

图 4(a) 展示了发展中地区和发达国家页面的 JavaScript 处理时间累积分布函数。红色曲线表示发展中地区页面的累积分布函数，其中实线表示原始页面的结果，虚线表示 *SlimWeb* 解决方案的结果。同样，发达国家页面的结果以绿色显示，实线表示原始页面，虚线表示*SlimWeb*。结果显示，与发达国家页面相比，手机浏览器处理发展中地区页面所需的 JavaScript 时间更长。这是因为在设计和构建发展中地区的页面时存在额外的低效之处：开发人员往往会误用实现方式，添加许多不必要的 JavaScript 库和 fi文件。图 4(b) 展示了这两类页面以及 *SlimWeb* 页面的平均 JavaScript 处理时间，误差线表示 95% 置信区间。结果显示，发展中地区页面处理 JavaScript 代码的平均时间约为 10 秒，而发达国家页面约为 7 秒。还可以看到，发展中地区页面的误差线往往大于发达国家页面，这表明发展中地区各页面的处理时间波 fl动更大。最后， fi图还显示，使用 *SlimWeb*这类解决方案（以点状阴影表示）确实能显著缩短两类页面的处理时间；与原始版本相比，处理时间约减少 40%。

页面加载时间累积分布函数和平均结果如图 4(c) 和 4(d) 所示。与 JavaScript处理时间结果类似，发展中地区网页的页面加载时间高于发达国家。然而，使fi用诸如 *SlimWeb* 这样的解决方案似乎对发达国家网页的益处略高于发展中地区网页：发达国家网页的页面加载时间减少了 30%，而发展中地区网页减少了 22%。

6 讨论与结论

尽管用于应对网页复杂性的解决方案在加快页面加载和节省数据套餐方面取得了显著改进，但缺乏用于比较分析的统一框架阻碍了这些解决方案的适用性。我们的框架能够快速且一致地评估网页复杂度解决方案，从而朝着根据用户需求和偏好促进发展中地区解决方案选择决策的方向迈出重要一步。我们设想了三种使用场景：1）研究人员旨在分析其提出的解决方案与现有方法相比的性能，以从不同视角全面呈现其优势和不足。2）开发人员旨在决定采用某种解决方案（或解决方案组合），以满足页面内容（内容丰富型、高度交互型、阅读模式）以及不同目标用户群体的要求。3）如果开发人员针对某个网页提供了多个版本，用户可以根据其设置（网络状况和设备）和偏好（加快页面加载或节省带宽）使用符合自身需求的页面版本。我们对现有网页复杂度解决方案生成的 1000 个网页的五个版本进行了分析，这些解决方案涵盖不同类别。分析表明，页面加载速度的显著提升可能会增加 CPU消耗，并影响页面结构和交互性。

未来，我们计划将页面组件的优化与处理加速相结合，以应对如何在保持页面结构和交互功能的同时，最大限度提升时序和能耗收益这一挑战。此外，我们计划将 Wasef 开源，并提供一个在线测试平台，配备多台服务器、存储单元以及不同类型的手机，供研究社区提交和运行定制实验，并集成他们自己的网页复杂性解决方案进行评估。我们希望 Wasef 能够助力改善网页浏览

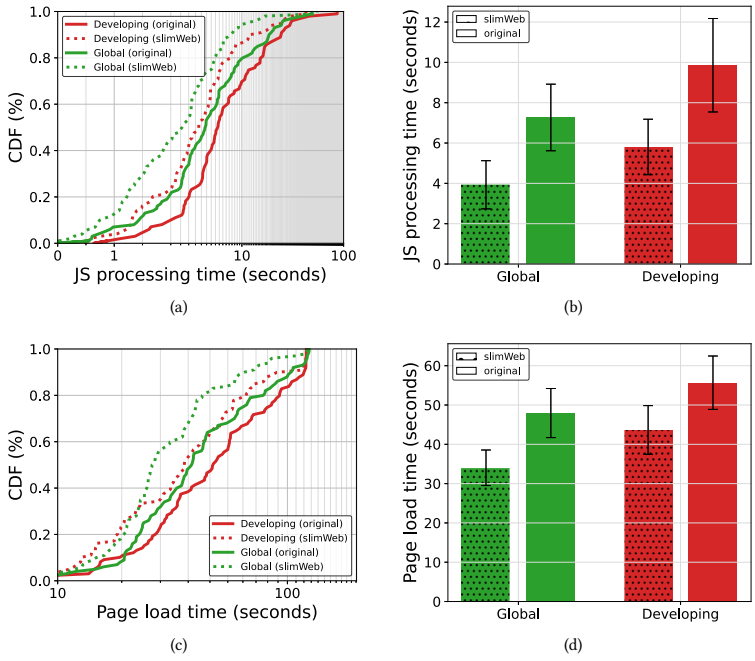


图 4. Q

体验，促进发展中国家获得更好的网页浏览体验，并帮助人们更深入地了解这些国家网页的复杂性。

参考文献

[1] [未知日期]。《第三世界152个发展中国家名单》。于2021年10月6日取自 https://www.worlddata.info/developing-countries.php[2] 2008。于2021年6月21日取自 https://www.semrush.com/siteaudit/[3] 2016。使用手机服务的非洲人多于使用自来水的人。https://edition.cnn.com/2016/01/19/africa/africa-afrobarometer-infrastructure-report/index.html。访问日期：2020年10月11日。[4] 2019。面向非洲的20美元手机是MWC最出人意料的主角。https://thenextweb.com/plugged/2019/02/26/a-20-phone-for-africa-is-mwcs-unlikeliest- hero/。访问日期：2020年10月11日。[5] 2020。推出全球最实惠的ff智能功能手机——The Digit 4G。https://www.kaiostech.com/introducing-the-worlds-most- affordable-smart-feature-phone-the-digit-4g/。访问日期：2020年10月11日。[6] 2020。据报道，Jio可能于2020年12月前在印度推出价格实惠的ffAndroid手机。https://www.bgr.in/news/jio-launch-india-android-phone-low- cost-2020-price-more-913657/。访问日期：2020年10月11日。[7] 2020。让移动互联网技术更加实ff惠。https://www.newpath.com/。访问日期：2020年10月11日。[8] 2020。六家新合作伙伴将在非洲推出运行KaiOS的实ff惠智能功能手机。https://www.kaiostech.com/press/six-new-partners- to-deliver-affordable-smart-feature-phones-running-on-kaios-in-africa/。访问日期：2020年10月11日。[9] 2021。全球浏览器市场份额 | StatCounter 全球统计。于2021年6月24日取自 https://gs.statcounter.com/browser-market-share[10] 2021。免费的网站速度测试。https://www.uptrends.com/tools/website-speed-test。访问日期：2021年6月21日。[11] 2021。全球有多少智能手机。https://www.bankmycell.com/blog/how-many-phones-are-in-the-world。访问日期：2021年6月23日。[12] 2021。Lighthouse 性能评分。https://developers.google.com/web/tools/lighthouse。访问日期：2021年6月21日。[13] 2021。2020年移动设备与桌面设备使用情况。https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage。访问日期：2021年6月23日。[14] 2021。页面速度监控。https://www.pingdom.com/product/page-speed。访问日期：2021年6月21日。

[15] 2021。抵御操纵、面向研究的顶级网站排名。于2021年6月6日取自 https://tranco-list.eu/[16] Victor Agabagov、Michael Buettner、Victor Chudnovsky、Mark Cogan、Ben Greenstein、Shane McDaniel、Michael Piatek、Colin Scott、Matt Welsh 和 Bolian Yin。2015年。Flywheel：Google 面向移动网络的数据压缩代理。载于第12届{USENIX}网络系统设计与实现研讨会（{NSDI} 15）。367–380。[17] Web Almanac。2020年。Lighthouse 性能评分。https://almanac.httparchive.org/en/2020/performance。访问日期：2021年6月23日。[18] Waqar Aqeel、Balakrishnan Chandrasekaran、Anja Feldmann 和 Bruce M Mags。2020年。关于落地页和内部网页：网页性能测量中化身博士的奇特案例。载于 ACM 互联网测量大会论文集。680–695。[19] Luciano Baresi、William G Griswold、Grace A Lewis、Marco Autili、Ivano Malavolta 和 Christine Julien。2020年。移动领域软件工程的趋势与挑战。IEEE Software 38，1（2020年），88–96。[20] Enrico Bocchi、Luca De Cicco 和 Dario Rossi。2016年。测量网页用户的体验质量。ACM SIGCOMM《计算机通信评论》46，4（2016年），8–13。[21] Timm Boettger、Ghida Ibrahim 和 Ben Vallis。2020年。互联网如何应对新冠肺炎：来自 Facebook 边缘网络的视角。载于 ACM 互联网测量大会论文集。34–41。[22] Catchpoint Systems 公司。2020年。WebPageTest——网站性能和优化测试。于2021年6月21日取自 https://www.webpagetest. org/[23] Moumena Chaqfeh、Rohail Asim、Bedoor AlShebli、Muhammad Fareed Zaffar、Talal Rahwan 和 Yasir Zaki。2023年。迈向没有数字不平等的万维网。《美国国家科学院院刊》120，3（2023年），e2212649120。[24] Moumena Chaqfeh、Russell Coke、Jacinta Hu、Waleed Hashmi、Lakshmi Subramanian、Talal Rahwan 和 Yasir Zaki。2022年。JSAnalyzer：通过消除非关键 JavaScript 简化移动网页的网页开发者工具。ACM《网络汇刊》16，4（2022年），1–31。[25] Moumena Chaqfeh、Muhammad Haseeb、Waleed Hashmi、Patrick Inshuti、Manesha Ramesh、Matteo Varvello、Fareed Zaffar、Lakshmi Subramanian 和 Yasir Zaki。2021年。拦截还是不拦截：通过 JavaScript 分fi类即时加速移动网页。arXiv 预印本 arXiv:2106.13764。[26] Moumena Chaqfeh、Yasir Zaki、Jacinta Hu 和 Lakshmi Subramanian。2020年。JSCleaner：通过 JavaScript 清理减少移动网页杂乱。载于2020年万维网大会论文集。763–773。[27] maximilianhils cortesi 和 raumfresser。2019年。mitmproxy——交互式 HTTPS 代理。https://mitmproxy.org/。访问日期：2019年10月13日。[28] Mallesham Dasari、Santiago Vargas、Arani Bhattacharya、Aruna Balasubramanian、Samir R Das 和 Michael Ferdman。2018年。设备性能对移动互联网体验质量的影响。载于2018年互联网测量大会论文集。1–7。[29] Steven Englehardt 和 Arvind Narayanan。2016年。在线跟踪：对100万个网站的测量与分析。载于2016年 ACM SIGSAC 计算机与通信安全大会论文集。ACM、Springer 出版公司，有限公司，奥地利维也纳，1388–1401。[30] Anja Feldmann、Oliver Gasser、Franziska Lichtblau、Eric Pujol、Igmar Poese、Christoph Dietzel、Daniel Wagner、Matthias Wichtlhuber、Juan Tapiador、Narseo Vallina-Rodriguez 等。2021年。新冠肺炎疫情对互联网流fi量的影响。载于德国宽带覆盖；第15届 ITG 研讨会。VDE，1–5。[31] Mohammad Ghasemisharif、Peter Snyder、Andrius Aucinas 和 Benjamin Livshits。2019年。SpeedReader：让阅读模式更快速、更私密。载于万维网大会。526–537。[32] Waleed Hashmi、Moumena Chaqfeh、Lakshmi Subramanian 和 Yasir Zaki。2020年。PQual：自动化网页定性评估。载于第33届 ACM 用户界面软件与技术研讨会（UIST）的附加出版物（虚拟会议，原定于美国明尼苏达州明尼阿波利斯举行）（UIST '20）。计算机协会，美国纽约州纽约。[33] Waleed Hashmi、Moumena Chaqfeh、Lakshminarayanan Subramanian 和 Yasir Zaki。2022年。QLUE：用于统一定性评估网页的计算机视觉工具。载于2022年 ACM 万维网大会论文集（虚拟会议，法国里昂）（WWW '22）。计算机协会，美国纽约州纽约，2400–2410。https://doi.org/10.1145/3485447.3512112[34] Tobias Hoβfeld、Florian Metzger 和 Dario Rossi。2018年。速度指数：将用户感知网页性能的行业标准与网页体验质量联系起来。载于2018年第十届多媒体体验质量国际会议（QoMEX）。IEEE，1–6。[35] J ason Huggins。2019年。Selenium WebDriver。浏览器自动化。https://www.seleniumhq.org/projects/webdriver/。访问日期：2019年5月14日。[36] Umar Iqbal、Zubair Shafiq 和 Zhiyun Qian。2017年。广告拦截之战：反广告拦截过滤列表的回顾性测量与分析。载于2017年互联网测量大会论文集。ACM，171–183。 fiJordan Jueckstock 和 Alexandros Kapravelos。2019年。VisibleV8：对真实环境中 JavaScript 的浏览器内监控。载于互联网测量大会论文集。393–405。[37] Andrew J. Kaizer 和 Minaxi Gupta。2016年。迈向基于机器的 JavaScript 导向跟踪的自动识别。载于2016年 ACM 国际安全与隐私分析研讨会论文集（美国路易斯安那州新奥尔良）（IWSPA '16）。计算机协会，美国纽约州纽约，33–40。https://doi.org/10.1145/2875475.2875479[38] Conor Kelton、Matteo Varvello、Andrius Aucinas 和 Benjamin Livshits。2021年。BrowseLite：面向网络的私密数据节省解决方案。arXiv 预印本 arXiv:2102.07864（2021年）。fiJesutofunmi Kupoluyi、Moumena Chaqfeh、Matteo Varvello、Russell Coke、Waleed Hashmi、Lakshmi Subramanian 和 Yasir Zaki。2022年。Muzeel：评估 JavaScript 死代码消除对移动网页性能的影响。载于第22届 ACM 互联网测量大会论文集。335–348。

[41] Adam Lerner、Anna Kornfeld Simpson、Tadayoshi Kohno 和 Franziska Roesner。2016年。《Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996年 to 2016年》。载于第25届 USENIX 安全研讨会（USENIX 安全 16）。[42] Timothy Libert。2015年。《揭示隐藏网络：对 100 万个网站第三方 HTTP 请求的分析》。arXiv 预印本 arXiv:1511.00619（2015年）。[43] Ivano Malavolta、Eoin Martino Grua、Cheng-Yu Lam、Randy de Vries、Franky Tan、Eric Zielinski、Michael Peters 和 Luuk Kaandorp。2020年。《在 Android 设备上自动执行基于测量的实验框架》。载于第35届 IEEE/ACM 自动化软件工程国际会议研讨会（ASEW '20）。ACM。https://github.com/S2-group/android-runner/blob/master/documentation/A_Mobile_ 2020.pdf[44] Matthew Malloy、Mark McNamara、Aaron Cahn 和 Paul Barford。2016年。《广告拦截器：全球普及程度与影响》。载于《2016年互联网测量大会论文集》。ACM，119–125。[45] Usama Naseer、Theophilus A Benson 和 Ravi Netravali。2021年。《WebMedic：化解下一十亿移动网页用户面临的内存与功能性之间的矛盾》。载于《第22届移动计算系统与应用国际研讨会论文集》。71–77。[46] Javad Nejati、Meng Luo、Nick Nikiforakis 和 Aruna Balasubramanian。2020年。《移动速度需求：移动网页性能的历史分析》。[47] Ravi Netravali、Ameesh Goyal、James Mickens 和 Hari Balakrishnan。2016年。《Polaris：通过细粒度依赖跟踪加快页面加载》。载于第13届 USENIX 网络系统设计与实现研讨会（NSDI 16）。USENIX 协会，圣克拉拉，加利福尼亚州。https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/netravali[48] Ravi Netravali 和 James Mickens。2018年。《Prophecy：利用最终状态写入日志加速移动页面加载》。载于第15届 USENIX 网络系统设计与实现研讨会（NSDI 18）。USENIX 协会，伦顿，华盛顿州，249–266。https://www.usenix.org/conference/nsdi18/ presentation/netravali-prophecy[49] Ravi Netravali、Vikram Nathan、James Mickens 和 Hari Balakrishnan。2018年。《Vesper：测量网页的可交互时间》。载于第15届 USENIX 网络系统设计与实现研讨会（NSDI 18）。USENIX 协会，伦顿，华盛顿州，217–231。https://www.usenix.org/conference/nsdi18/presentation/netravali-vesper[50] Niels Groot Obbink、Ivano Malavolta、Gian Luca Scoccia 和 Patricia Lago。2018年。《驾驭 JavaScript 死代码消除挑战的可扩展方法》。载于《2018年 IEEE 第25届软件分析、演化与再工程国际会议》（SANER）。IEEE，291–401。[51] Sergio Pastrana、Alice Hutchings、Daniel Thomas 和 Juan Tapiador。2019年。《测量网络性交易》。载于《互联网测量大会论文集》。463–477。[52] Peng Peng、Limin Yang、Linhai Song 和 Gang Wang。2019年。《揭开 VirusTotal 黑箱：分析在线网络钓鱼扫描引擎》。载于《互联网测量大会论文集》。478–485。[53] Victor Le Pochat、Tom Van Goethem、Samaneh Tajalizadehkhoob、Maciej Korczyński 和 Wouter Joosen。2018年。《Tranco：经加固以防操纵、面向研究的顶级网站排名》。arXiv 预印本 arXiv:1806.01156（2018年）。[54] Pijush Kanti Dutta Pramanik、Nilanjan Sinhababu、Bulbul Mukherjee、Sanjeevikumar Padmanaban、Aranyak Maity、Bijoy Kumar Upadhyaya、Jens Bo Holm-Nielsen 和 Prasenjit Choudhury。2019年。《功耗分析、测量、管理及相关问题：智能手机电池与能源使用的最先进综述》。IEEE Access 7（2019年），182113–182172。[55] Hispar 项目。2020年。Pagetypes 数据集、Hispar 列表工具及档案。检索于2021年6月6日，网址：https://hispar.cs.duke.edu/[56] Vaspol Ruamviboonsuk、Ravi Netravali、Muhammed Uluyol 和 Harsha V Madhyastha。2017年。《Vroom：借助服务器辅助依赖解析加速移动网络》。载于《ACM 数据通信特别兴趣小组会议论文集》。390–403。[57] Iskander Sanchez-Rola 和 Igor Santos。2016年。《已知与未知：通用网页跟踪分析器对 100 万个网站的研究》。技术报告。德乌斯托大学 DeustoTech 技术报告。[58] Grant Storey、Dillon Reisman、Jonathan Mayer 和 Arvind Narayanan。2017年。《广告拦截的未来：分析框架与新技术》。arXiv 预印本 arXiv:1705.08568（2017年）。[59] Elena Terenteva。2018年。《网站性能研究》 [《SEMrush 研究》 2018]。检索于2021年6月21日，网址：https://www.semrush.com/blog/website- performance-semrush-study/[60] Phani Vadrevu 和 Roberto Perdisci。2019年。《所见非所得：发现并跟踪社会工程攻击活动》。载于《互联网测量大会论文集》。308–321。[61] Matteo Varvello、Jeremy Blackburn、David Naylor 和 Konstantina Papagiannaki。2016年。《Eyeorg：众包网页体验质量测量平台》。载于《第12届新兴网络实验与技术国际会议论文集》。399–412。[62] Xiao Sophia Wang、Aruna Balasubramanian、Arvind Krishnamurthy 和 David Wetherall。2013年。《借助 WProf 揭示页面加载性能的奥秘》。发表于第10届 USENIX 网络系统设计与实现研讨会（NSDI 13）。USENIX，隆巴德，伊利诺伊州，473–485。https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/wang_xiao[63] Xiao Sophia Wang、Arvind Krishnamurthy 和 David Wetherall。2016年。《借助 Shandian 加快网页加载》。载于第13届 USENIX 网络系统设计与实现研讨会（NSDI 16）。USENIX 协会，圣克拉拉，加利福尼亚州，109–122。https://www.usenix.org/conference/ nsdi16/technical-sessions/presentation/wang[64] Jihwan Yeo、Changhyun Shin 和 Soo-Mook Moon。2019年。《在非确定性 JavaScript 执行下基于快照的网页应用加载加速》。载于《万维网大会》（美国加利福尼亚州旧金山）（WWW '19）。计算机协会，纽约，纽约州，2215–2224。https://doi.org/10.1145/3308558.3313575