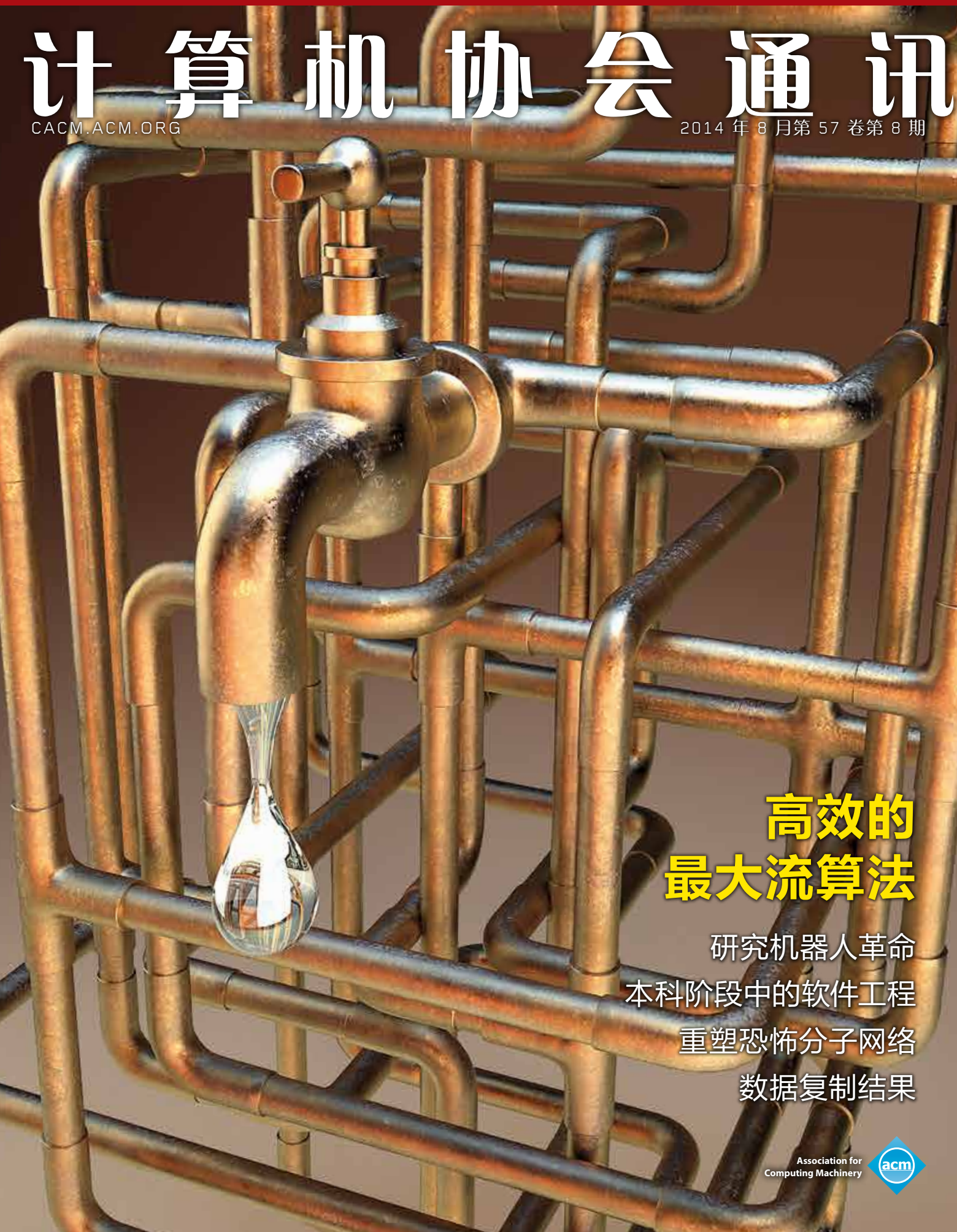
★CACM 中国版★

计算机协会通讯

CACM.ACM.ORG

2014 年 8 月第 57 卷第 8 期



高效的最大流算法

研究机器人革命

本科阶段中的软件工程

重塑恐怖分子网络

数据复制结果

Association for Computing Machinery

8/2014
第 57 卷 第 8 期

观点

实践

关于封面：
Andrew Goldberg 和 Robert Tarjan 在本期的封面故事中探索最大流算法背后的技术。（第 82 页）。作者认为完全理解最大流问题的道路还很长，而人们也在继续发现和改进算法。封面插图由 Jurgen Ziewe 提供。

从左到右：图片来源：SIMON DAWSON/BLOOMBERG/GETTY IMAGES; RAUL ARBOLEDA/AFP/GETTY IMAGES; 插图由 JURGEN ZIEWE 提供

投稿文章

评论文章

研究亮点

Association for Computing Machinery
Advancing Computing as a Science & Profession

DOI:10.1145/2633039

Frank Levy 和 Richard J. Murnane

观点

研究机器人革命

考虑通过计算机科学家与经济学家的跨学科研究计划，探究计算机对工作的影响。

1960 年，科学家发现大气中的 CO_2 浓度在不断上升。尽管知识在进步，但人们对全球变暖的认识即便在十年后也十分模糊。科学史学家 Spencer Weart 对这一状况进行了总结。“20 世纪 70 年代早期，随着环境保护主义的兴起，越来越多的人开始怀疑人类活动对地球是否有益。对气候的好奇变成了焦虑。一些科学家指出，人类活动除了产生温室效应以外，还向大气排放灰尘和烟雾颗粒，从而阻断阳光，使地球变冷。此外，北半球的气候统计分析表明，变冷的趋势从 20 世纪 40 年代就已经开始。大众媒体（在他们涉及这一问题的有限程度上）非常困惑，时而预测全球变暖，冰帽融化，沿海地区被洪水淹没，时而又告诫人们一个灾难性的新冰河时代即将来临。”[a]

a 请参阅 http://www.aip.org/history/climate/summary.htm。

目前我们对机器人革命的了解大约相当于上世纪 70 年代对全球变暖的了解。我们清楚地知道一些事情。计算机消除了制造业、行政工作以及其他“中等技能”领域中涉及结构化任务的工作。计算机在某些领域，特别是为技术娴熟的人员创造了就业机会。与全球变暖不同，计算机化工作应会增加 GDP。除开这些事实，剩下的就是无尽的想像。我们不太了解机器人革命对

图片由 SIMON DAWSON/BLOOMBERG/GETTY IMAGES 提供

英国 Argos 商品配送中心一名员工监控自动化生产线。

就业和工资到底有多大影响，就如同我们不大了解革命的进展速度一样。正如 Weart 可能会说的那样，大众媒体（以及我们当中的其他人）非常困惑。

困惑的原因之一是对这个问题缺乏持续的深入研究工作，类似那种有助于我们最终认清全球变暖现象的研究。鉴于迅速发展的计算机化工作可能产生的颠覆性后果，这项研究计划非常有必要。以下是关于如何推进这项研究计划的一些想法以及必须克服的障碍。我们的发现部分来自过去两年我们在麻省理工学院的 10 次午餐会经历，当时经济学家和计算机科学家们齐聚一堂，相互切磋。

正如全球变暖研究涉及多个学科一样，研究机器人革命需要从计算机科学家和经济学家的跨学科学习开始。有关机器人革命的论文还比较少，部分原因是大多数经济学家对人工智能知之甚少，而计算机科学家的经济学知识又相对欠缺。

让我举两个例子加以说明。早在麻省理工学院的午餐会上，当计算机科学家说到“计算机缺乏常识”时，一位经济学家询问这是什么意思。计算机科学的教授非常清晰地解释了“常识问题” — 人类执行任务时用到的成千上万看似微不足道但软件难以捕获的事实。那次午餐会上共有 6 名经济学家，其中有 5 名（包括我们）曾就计算机对工作的影响发表过长篇大论。但据合理推断，最多只有一名经济学家曾听说过常识问题。而没有人在其著作中提到过这个问题。

跨学科研究的主要障碍是非对称的激励结构。

同样，在最近的 *ACM 通信* 的观点栏目中，软件开发者 Martin Ford 讨论了机器人革命可能带来的一种影响：“入门级职位尤为脆弱，所以实际上在过去十年中，新大学毕业生的工资一直在下降，而多达 50% 的新毕业生被迫从事不需要大学文凭的工作。”[3]

经济学家认为推断因果关系没有那么简单。过去十年中，计算机的应用已延伸到经济领域，如 Ford 所说，新大学毕业生的工资也一直在下降。但过去十年中还发生了其他事情，包括 2008 年金融危机和随后的经济大衰退。历史表明，金融危机之后的经济衰退平均要持续五年[4]，并且所有经济衰退的影响对劳动市场的新进入者尤其巨大。要严谨地估算计算机对新大学毕业生的影响，必须先控制经济衰退的影响。

跨学科研究的主要障碍是非对称的激励结构。经济学家有偿研究影响劳动力市场的因素（包括计算机）。他们知道研究的资助者、供他们展示成果的研讨会和会议，以及供他们发表论文的期刊。据我们所知，研究机器人革命的劳动力市场影响的计算机科学家并没有这些基础设施。

相反，研究计算机化工作的劳动力市场影响占用了职业发展活动的时间。由于这种不对称，对技术发展的就业影响感兴趣的计算机科学家很可能分散在不同的机构，而不像在许多计算机系中的机器视觉组或自然语言处理组那样集中存在。稍后我们再回到这个话题。

有人可能会问另一个障碍是否是计算机科学家不愿意去考虑其工作可能对就业带来的负面影响。经济学家可以在这个问题上带来一些

安慰。在金融危机之后，经济学家们受到了强烈批评，因为他们提倡放松管制以及导致金融危机的金融工具，更不用说他们未能率先预测到金融危机。在实践中，经济学家们对这些批评视而不见[b]，而继续研究他们感兴趣的问题。

可以很容易构想出一项协作研究计划，从研究者之间的对话开始，然后进行案例研究，从而先后产生假设、进行计量经济分析和模拟 — 这些研究使未来更清晰。问题是如何开始。更确切地说，问题是如何识别和连接想要从事这项工作的人。

识别感兴趣的经济学家很简单。几乎所有从事机器人革命的经济学家都属于美国全国经济研究所 (NBER) 的三个计划组（劳动研究、教育经济学以及生产力、创新和创业），并可通过 NBER 组织迅速与他们取得联系。识别感兴趣的计算机科学家则更为困难，因为正如我们讨论的那样，感兴趣的个人很可能分散在世界各地，而 且没有与之直接相关的 ACM 特别兴趣组。我们需要发送一枚照明弹，以便感兴趣的计算机科学家响应，并能够与其他计算机科学家和志趣相投的经济学家联系。

此“照明弹”在逻辑意义上可以是 *ACM* 主办的计算机化工作的就业影响（正面的或负面的）专题讨论会。可通过 *ACM 通信* 和其他 ACM 渠道将摘要征集信息传达给计算机科学家。可通过 NBER 内部的项目电子邮件组将摘要征集信息传达给经济学家。

专题讨论会有两个目标：作为感兴趣的研究人员的联络点；开始为全新的重要领域制定标准。因此，专题讨论会的目标应该是推动研究，而不是妄加猜测。我们目前仍处于起步阶段，虽然现在期待计算机科学家与经济学家之间的协作研究还为时尚早，但我们至少可以要求提交材料贴合作者的专业知识，同时注重近期（或过去）发展，而不是放眼遥远的未来。

对技术发展的就业影响感兴趣的计算机科学家很可能分散在不同的机构。

对于经济学家而言，专业对口研究的一个例子就是计量经济研究，旨在估计一项或多项技术发展的就业或工资影响。[1] 这项研究是分析历史，而非预测未来，但会以有助于未来开展协同工作的方式，呈现经济学家的建模和数据处理技术。另一个例子是对特定技术的实施进行案例研究，为未来的工作提出假设[2]。

对于计算机科学家来说，相关研究的一个例子就是对充分发展后可能对劳动力市场产生重大影响的技术进行剖析。剖析包括讨论充分发展的障碍，同时作者对这些障碍的未来发展进行评估。要实现全自动汽车，必须解决哪些导航和识别问题？ ‘要在应答特定领域的客服电话方面优于国外呼叫中心，自然语言处理和信息检索技术还需要哪些改进？

基于这些例子，需选出一个计算机科学家和经济学家的甄选委员会，以编写摘要征集信息，同时评选递交材料。递交的论文将由一名计算机科学家和一名经济学家讨论。如果修订后的论文质量合格，可在 *ACM 通信* 上发表精简版，完整版可发布在相关网站或其他刊物上。专题讨论会和网站也都可以作为交会点，供感兴趣的计算机科学家和经济学家发布联系信息和兴趣，并开始相互通信，为协作工作打下基础。据非正式谈话，有人对资助这类会议非常感兴趣。

在接下来的 25 年，全球变暖和计算机化工作的普及都会给社会带来意义深远的改变。研究工作早就该开始显露同样巨大的力量。 C

b 例如，请参考 MIT 150 [周年]庆上的“金融技术的发展”专题讨论会：http://mit150.mit.edu/symposia/economics#video。

c 今年 7 月，加州伯克利举行了有关这些主题的研讨会，作为 2014 机器人：科学和系统会议的一部分。

参考资料

1. Autor, D.H., Dorn, D. and Hanson, G.H.Untangling trade and technology:Evidence from local labor markets. Working Paper, Department of Economics, MIT (Mar. 2013); http://economics.mit.edu/files/8763.
2. Autor, D.H., Levy, F. and Murnane, R.J.Upstairs, downstairs:Computers and skills on two floors of a large bank.*Industrial and Labor Relations Review 55*, 3 (Apr. 2002), 432–437.
3. Ford, M. Could artificial intelligence create an unemployment crisis?*Commun.ACM 56*, 7 (July 2013); DOI:10.1145/2483852.2483865.
4. Reinhart, C.M. and Rogoff, K. *This Time Is Different:Eight Centuries of Financial Folly.*Princeton University Press, 2009.

Frank Levy (flevy@MIT.EDU) 是麻省理工学院（马萨诸塞州剑桥）的 Rose 冠名荣誉退休教授。

Richard J. Murnane (Richard_Murnane@Harvard.edu) 是哈佛大学教育研究生院的 Thompson 冠名教授。

译文责任编辑：陈文光

文章编写主导者 acmqueue
queue.acm.org

应对专业软件开发的需求

MICHAEL J. LUTZ，J. FERNANDO NAVEDA，和 JAMES R. VALLINO

本科阶段中的软件工程

在 1996 年的**秋季**学期中，罗彻斯特理工学院（RIT）启动了美国首个软件工程本科专业。[9,10] 经过为时五年的规划、发展和评审后，该专业终于达到了其顶峰，顺利启动；从一开始，该专业便设计为帮助学生打好基础，让他们毕业时能够胜任专业的商业和工业软件开发工作。

开班时，只有 15 名学生。不过，经美国工程与技术鉴定委员会（ABET）认证的这个专业已经稳步增长。如今，本科阶段的学生总数已然超过 400 名。实习生和毕业生获得了规模不同的多家组织的聘用，其中包括微软、谷歌、苹果和联合技术公司以及各种政府机构。该专业属于 RIT 中独立的软件工程系，具有必要的独立性和灵活性，可以在不断的进步中保证自己的健全性。

它的首要重点是培养专业的、实战型的软件工程师。这点在课程设置上体现得最为直接和明显：在两年的基础课程完成后，需要进行两年的合作教育。学生交替地感受正规教育与带薪职业经历；当为期五年的专业教育结束时，学生不仅拥有了扎实的学科基础知识，还获得了重要的实践经验。这些毕业生的需求量很大，因为他们已经准备充分，能够定义、设计、开发和交付高质量的软件系统。

当然，人们仍然抱有疑问：为什么要设立专门的软件工程学位？毕竟，大多数工业领域的新进员工来自计算机科学和工程中的传统专业。在本文中，我们阐释了开创新方向的根本理由——我们坚信，业界需要初级软件工程师；我们确信，我们能够提供一种教育经历，让学生准备得更充分，能更好地胜任软件领域的职业。接下来，我们仔细分析了 RIT 的专业与典型的计算机

照片由 GOODLUZ PRODUCTIONS 提供

科学本科专业之间的差异。这反过来又引发我们阐述了教学法以及软件工程在计算机科学课程体系中的状况。随后，我们讨论了 RIT 的专业与业界的关系以及实习生和毕业生的培养方法。

动机

在二十世纪八十年代后期，本文的作者之一（Lutz）从 RIT 离职两年以从事产业方面的工作：他首先去了 GCA/Tropel，一家光学计量产品的制造商；然后又去了 Eastman Kodak，在那儿他带领多个团队开发嵌入式系统和应用级软件。他的职责包括面试、招聘和指导新进的大学毕业生。不过，在此期间，他看到了一些令人不安的现象。从总体上来看，这些毕业生在基本计算理论和技术方面的基础扎实。很多人修了算法分析和计算理论的课程，而且多数人也接触过操作系统、程序设计语言概念、人工智能、图形学和编译器设计。不过，他们却缺乏必要的实践背景，不能有效地在大型复杂工业质量系统上开展工作。

特别是，这些毕业生只有一丁点（如果有）作为软件团队成员的工作经历。但是，软件团队工作却在业界的实践中相当普遍。他们的设计知识往往局限于计算机科学家感兴趣的构件——编译器、操作系统、图形库以及其他的东西——但他们却几乎没有理解设计本身就是一项活动。多数人没有版本控制的经验，熟悉配置管理的就更少。他们的测试知识通常相当贫乏，几乎没人听说过验证和校验。最后，除了死记硬背瀑布模型之外，他们几乎或完全不知道创建产品时涉及的实际过程。

与在业界和软件工程教育的其他人交谈后，我们发现这些问题极为普遍。这个问题是科学与工程对立的老问题：科学的目标是增加和拓展知识；而工程的目的是应用这类知识创造有用的产品。在传统上，这被表述为科学家“为了学习而创造”，而工程师“为了创造而学习”。在计算机科学和软件工程之间，看起来恰恰适于以这种方式加以区分，正如过去物理学和工程物理之间出现的差异一样。

当我们在 RIT 开发软件工程的课程体系时，很多大学已经提供了软件工程方面的硕士课程。当时流行的观点是，本科生应该继续攻读计算机科学学位，然后就读硕士来完成他们的教育。但是，由于大多数计算机科学的毕业生在毕业后直接进入业界，而且很多人可能永

远都不会完成 MSSE（软件工程硕士），所以这个方法存在问题。让我们看看下面的例子：教新驾驶员学驾驶的方法之一是，先讲内燃机、传动系统和电气系统的理论，然后把车钥匙交给驾驶员，让他开着车去兜风；在新驾驶员撞了几根电杆，撞坏一些邮箱后，教练再说“现在你可以学怎么开车了。”从我们的角度来说，这与培养软件开发人员的 BSCS（计算机科学本科学位）/ MSSE 方法类似。

当然，必须有一些权衡和取舍。正如机械工程师掌握的物理学的深度不需要与物理学专业的学生一样，软件工程师需要掌握的计算机科学的深度也不需要与计算机科学专业的学生一样。不过，我们的论点是，在进行当代的软件开发实践时，软件工程知识能弥补科学知识深度的不足。本文中，我们探讨了计算科学与软件工程之间的差异，并将其作为揭示这些权衡和取舍的一种途径。

软件工程教育的宣言

2001 年，一群杰出的软件工程专家发表了《敏捷软件开发宣言》[2]。他们阐述了软件开发中不同方法的各种利弊，比如“应对变化”和与之对比的“紧跟计划”。他们得出结论并声明：虽然所有这些方法都存在价值，但是他们提倡胜过其他方法的“敏捷”方法。

与此类似，我们并没有忽视传统计算机科学的方法的价值；在软件工程师的教育中，我们只是把其他方法看得更重。从敏捷宣言获取灵感后，我们阐述了在 RIT 的软件工程专业中使用的相关方法和权衡。虽然我们承认某些计算机科学专业和教员确实采用了我们所提出的一种或多种方法，但是我们却发现其他专业未能达到我校该专业在这方面的实施程度。关于 RIT 的专业的更多详情，请参阅课程图。[12]

水平课程与垂直课程的对比它们在哲学层面上的最大区别或许是，在工程知识的广度（水平的）与特定的技术专长的深度（垂直的）之间的取舍不同。RIT 的软件工程专业毫不掩饰地基于前者。在必修课中，学生多次反复体验了开发生命周期的全过程，其中涵盖从客户需求到产品交付以及中间的所有活动（例如，正式和非正式的建模；架构和设计；测试和质量保证；规划、估算和跟踪；过程和项目管理）。当然，也有单独的课程专注于在工程师的专业职责中存在的某些特定方面；不过，到他们着手实案毕业班项目时，学生们已经掌握了从启动项目到完成项目所需的背景知识。

实际上，毕业班项目说明了水平的方法。在众多计算机科学专业中，软件工程局限于一个学期的项目，但 RIT 的软件工程专业的学生却拥有两个学期的时间开发基于团队的毕业班项目，用于彰显他们之前学到的所有知识。团队会面对真正的客户（产业客户、非盈利机构或是 RIT 的内部机构），他们负责确定项目范围，协商需求，并在存在约束的条件下设计解决方案（例如，与现有系统兼容），实施风险分析以及编制和实施合适的开发计划。学生开发的很多项目已经在项目赞助人的现场上线运营，这证实了这个方法是成功的。

团队协作与个人活动的对比很多计算机科学课程对个人能力的重视程度超过团队协作。但是，在团队中解决问题却是 RIT 的软件工程专业的一个特征。实际上，除了两门课程以外（有关个人软件工程的课程和形式化数学建模的课程），软件工程的所有其他课程均把团队项目作为评分的重要组成部分。

第二年的入门课程（软件工程、计算机科学和计算机工程专业的入门课程）则把团队协作提升为专业实践的基础。特定的角色和职责得以强调，团队凝聚力、冲突解决以及基于团队的成功也得以凸显。该课程还确保学生能够接触版本控制，因为版本控制不仅在记录团队成员的贡献时必不可少，在检测和解决对文档和源代码的冲突变更时也至关重要。

对于依照团队活动评出的成绩（约 50%），团队作为一个整体得分。另外，还会根据每个团队成员对团队的贡献、教员的观察、版本控制的日志和保密的同学互评为每个成员评分。

入门课程是计算机科学和计算机工程专业都必修的唯一课程。不过，对于软件工程师来说，它只是很多课程的开始。学校希望学生组成四到六个人的团队一起工作，并把它作为职业教育不可分割的一部分。通过这种专业课程，典型的软件工程学生能够参与 20 多个不同的团队。

设计和建模与程序设计和编码的对比 RIT 的软件工程的学生对程序设计本身 的接触主要来自于延续性的基础性计算机科学课程。虽然基础性的软件工程和高级软件工程的课程均探讨了程序设计的技术，但其并不是重点。与此相反，我们把程序设计的能力作为进入该领域的入场券——一种对会员的基本要求（如果你想入会的话）。在我们的大多数课程中，虽然各个团队需要编写软件系统的程序，但是我们的重点是把程序设计当成产品交付的一个必要环节。

下面的例子或许能帮助阐述这一点。虽然在学习计算机科学课程后，我们期望学生能够理解数据结构，并掌握复杂性的基本概念（比如“大 O”表示法），但我们极少要求他们从头开始构建此类数据结构。与此相反，我们强烈地鼓励他们在条件允许的情况下利用现有的组件。这些组件可能是语言（比如 Java 或 Ruby）的标准环境的一

部分，或是由第三方提供（例如，RubyGems）。团队需要尽职尽责，保证选择的组件呈现了某些质量属性，并提供了所需的功能；另外，他们还必须为使用的任何组件恰当地指出系属来源。

减少对程序设计 本身 的讲授为我们强调在建模和设计方面存在的更重要的问题留出了空间。[14] 在第二年的基础课中，除了之前讨论的团队协作部分之外，还有介绍基本的设计质量的部分，比如内聚与耦合、信息隐藏、按接口而不是实现进行设计，以及抽象成组件来提供定义明确的服务。对于主修此专业的学生而言，第二年的其他课程还提供了更多建模和设计方面的经历。

建模课程讲授了基于形式化的方法，用于建模，探索和验证设计。在使用了 Alloy[6] 和 Promela/Spin[5] 等工具后，学生学习了如何利用离散数学表示结构和行为属性以及如何使用相关的工具来验证有关系统总体属性的断言。除此之外，该课程提供了数据建模和关系型数据库理论的概述。在课程结束时，学生能够更好地理解严格的设计分析在软件系统分析中的作用。

子系统设计的课程明确地讲授了设计知识，它使用设计模式[4] 作为提升抽象层级的一种手段。我们的经验说明，通过给常用的结构和行为互动命名，并在设计练习中应用这些拓展后的词汇，学生开始远离实现细节，关注更高层级的组件关系。对于高年级的设计课程，无论这些课程讲授安全性、并发或是基于网络的系统，它们均能以此为基础，讨论设计理念和权衡。

并发和分布式系统设计的课程说明了我们与大多数计算机科学课程的另一种区别。虽然计算机科学通常会在操作系统或数据库系统中介绍这些问题，但是我们的课程更着重于并发和分布式的概念和问题本身，较少关注这些整体的人工构件。通常情况下，学生团队会设计、开发和交付一些把并发作为关键设计因素的系统，而不是数据库和操作系统，因此他们没有把并发作为只有专家才能涉足的领域。在多核计算机和云计算的时代里，这种方法对毕业生的帮助很大。

创建 RIT 的软件工程课程体系时，在我们的想法中，通过纪律严明的过程来教授专业精神的理念占有突出的地位。

纪律严明的过程与临时的开发的对比创建 RIT 的软件工程课程体系时，在我们的想法中，通过纪律严明的过程来教授专业精神的理念占有突出的地位。过程并不像很多人说的那样，是软件工程的终极目标，但它的确提供了一个框架来约束软件开发活动。在我们的课程体系中，过程是软件设计之外的另一个重要支柱。

然而，这并不是说，我们会把一个教条的方法强加在过程之上——实际上，我们确保学生熟悉多种过程方法，比如从严谨规划的过程到适应性更强的敏捷方法。[3] 选择适合于手上项目的过程并坚持到底相当重要，而认识到这一点则是高效的从业人员的素质之一。在应对不断变化的网络系统时，敏捷方法的收益颇大，但用于安全至上的场景（例如，飞行器的电传操纵系统）时，它反而会成为重大风险。

估算和跟踪则是在计算机科学专业中极少被探讨的重点过程部分。在个人软件工程第一年的课程中，学生们估算和跟踪了课堂活动和较长项目中的工作量。为了防止“做假”，评定学生成绩时，不仅评价了他们预测的准确程度，还考虑了他们对预测值和实际值会出现差异这一问题的反思。这种反思活动会缓慢地，但是确定地提升学生的估算能力，这也是在后面课程中进行团队估算的基础。

教学方法

在 RIT 的软件工程课程中，主动学习和基于团队的项目工作是教学法中两个最显著的特征。[8,13] 使用主动学

习的教育模式当然不是软件工程专业专有的，但是把它应用于整个课程体系却有点罕见。幸运的是，我们能够在教学设施中融入对它的支持。我们所有的课程几乎都在工作室风格的实验室中教授，而且我们还用课堂练习和团队项目活动取代了大量的讲授时间，让学生能够立即加深对课程概念的理解。在工作室风格的实验室中，每个座位上都配备了电脑，可以让学生在听课和个人或成对练习之间进行无缝转换。

每门课程均设置了贯穿整个学期的团队项目，而且该成绩占总成绩的至少 40%。为了支持课堂中和课堂外的团队活动，学校安排了 11 个七人团队室，每个团队室均配有白板、台式计算机和投影仪。团队室是工作室风格的实验室的扩展。在一堂典型的课程中，会有一半的时间放在工作室风格的实验室中进行讲授和课堂练习；其他的课堂时间则在团队室度过。

在团队室度过的时间里，学生或是组成随机的团队，一起使用课堂材料完成练习，或是与当前项目团队聚在一起，完成具体的项目工作。在此期间，教员会直接与每个团队互动，以便更好地了解团队以及团队成员的工作情况。团队有多次机会获取项目反馈，聆听设计指导。

设计这种教学法时，很多软件工程的教员回想起了他们业界生涯的开始阶段。那时候，他们接受的软件设计方面的教导来自指导他们的高级工程师。为团队分配的教员时间促进了那种互动。

随着时间的推移，我们意识到，团队甚至需要更多的支撑设施。与经验丰富的项目赞助人进行会谈、召开会议和评审时，团队室是理想的场所；但是，对于团队实施环节，它们还不够。为了克服这一缺陷，我们重新布置了工作室风格的实验室，把他们变成了软件工程协作实验室，其中包含了五个协作区，每个协作区可容纳六名学生。一台装在墙上的显示器显示了在四台桌下工作站中某台工作站或学生笔记本电脑的输出。工作站显示器的安装位置低于桌子，留出了宽敞的协作空间。几位在业界工作的人在参观后表达了他们对这种独特布置的赞赏，并希望在他们自己的团队区域内重建这一布置。

我们设立本科阶段的软件工程专业的动机是由于我们发现，初级软件开发人员所需的技能与计算机科学专业通常向学生教授的技能存在着不匹配。

计算机科学变了吗?

我们设立软件工程本科专业的动机是由于我们发现，初级软件开发人员所需的技能与计算机科学专业通常向学生教授的技能存在着不匹配。我们相信，20 年前描述的技能矛盾现在仍然存在。这种矛盾体现在《*2013 年计算机科学课程体系：计算机科学本科专业课程导则*》（CS2013），[7]，它是很多计算机科学专业的基础。

CS2013 的指导原则明确要求“课程必须……把专业实践（例如，交流技巧、团队协作、职业道德）作为本科教学的组成部分。计算机科学的学生必须学习把理论与实践相结合，认识抽象的重要性以及领略良好的工程设计的价值。他们期望计算机科学的毕业生拥有的特点之一是项目经验，其中“计算机科学的所有毕业生本应参与至少一个重要项目。在大多数情况下，这种经历将会是软件开发项目，但是在一些特殊的情况下，其他的经历也合适……学生应有机会发展他们的人际交往技巧，以此作为他们项目经验的一部分。”导则中最重要的这两点确认了这种对软件工程理念的需要。

《*ABET（美国工程与技术鉴定委员会）计算机专业认证准则*》是指导计算机科学专业课程内容设置的另一份文件。[1] 在题为“计算机科学以及类似名称的计算机专业的专业标准”中，对学生的学习成果规定如下：

学生学习该专业后，在毕业前必须能够获得：

（*j*）一种运用数学基础知识、算法原理和计算机科学理论的能力，在为基于计算机的系统进行建模和设计时，能够论证他们对设计选择中包含的各种权衡的理解。[CS]

（*k*）一种运用设计和开发原理的能力，能够构建复杂度各异的软件系统。[CS]

这两项成果明确地说明课程需要涵盖设计原理和开发实践，而这两者都在软件工程的范畴之内。不仅如此，我们还认为，（*j*）的建模和设计部分以及（*k*）的所有部分都属于软件工程。

既然 CS2013 导则和 ABET 认证要求已经确立了对软件工程的需求，现在让我们来了解下在该领域中现有的课程体系导则提供了哪些东西。CS2013 围绕技术确定了 18 个知识领域，如架构和组织、图形学和可视化、组网与通信、操作系统以及程序设计语言。只有三项——SD（软件开发基础）、SE（软件工程）、SP（社会问题和专业实践）——处于软件工程的范畴之内。导则中的评论确定 SE 和 SP 知识领域为特定的课程领域；在这些领域内，学生得以学习和实践团队协作和交流软技能。在这三个知识领域中，规定了软件工程方面的课题的最低课时为 SDF 10 小时，SE 28 小时，SP 1 小时。

即使学生可以把更多的时间花在作业和项目工作上，并在选修课上阅读更多的材料，这种最低标准也不足以培养初级软件工程师所需的全部的技能组合。当你考虑 SE 知识领域时，这点尤为正确。SE 有 14 页，是 CS2013 中最长的无交叉知识领域，它确定了 60 项核心课题，其中包含了 69 项学习成果；54 项选修课题，其中包含了 56 项学习成果。该知识领域的广度和深度引来了在软件工程教育的会议中可常听到的叹息。负责课程体系中软件工程的 CS 教员问道，“在我们的计算机科学课程中，我怎么才能把 SE 的核心课题和“软”性的团队协作和交流技巧放到同一门软件工程课程中去？“本科阶段的计算教育的现实是，虽然绝大多数学生有时间深入学习软件工程的课程，但是他们都没能全面地学习软件工程的课程。与此相反，他们就读计算机科学或计算机工程专业，然后他们只能从一门，通常也只有一门软件工程课程中学习软件工程技能。[15]

从产业的角度探讨软件工程教育

2013 年 5 月，本文的作者之一（Vallino）参加了罗切斯特 Java 用户组（Rochester Java Users Group）的一次会议。除了 Java 技术的某些方面的报告之外，该小组还举行了由 Bryan Basham 主持的软件教育 / 培训 / 认证方面的综合讨论。Bryan Basham 是一位活跃的开发人员，曾担任太阳微系统公司（Sun Microsystems）的 Java 培训师。对于讲授的知识与软件开发人员需要的技能组合之间存在的不匹配情况，他表达了自己的担忧。20 年前我们开始在 RIT 开发软件工程专业时，我们也怀有相同的见解。

接下来，在用户组会议中，参会者被要求回忆并列出他们在本科的课程作业中学到的东西。他们的列表确定了在传统计算机科学学位教育中探讨的技术领域。然后，参会者描述了他们认为一个合格的开发人员在软件开发活动中所需具备的技能。这一列表覆盖了 RIT 的软件工程专业的大多数要素，并要求很强的团队协作能力和交流技巧。这一经历加强了我们的观点，即软件工程专业需要应对专业软件开发的需要，至少由活跃的开发人员组成的参会者是这么认为的；另外，这个专业需要让更多的人了解，因为在参会者中，甚至没人听说过有大学开设了软件工程本科。

RIT 的软件工程专业的有效性可以依照对认证的自我研究进行量化评价。我们确实比较了计算机科学和软件工程的坊间指标。RIT 的职业服务办公室出版了薪酬数据[11]。与计算机科学和计算机工程以及 RIT 中所有其他的计算专业相比，软件工程的学生报告了每年最高的均小时实习工资以及全职工资的中位数。软件工程本科生的就业率在毕业时达到了 90% 以上。

另外，在实习评价的总结中，也给出了坊间证据证明了我们对学生的培训为他们的雇主创造了价值。某家航空公司工程部的经理聘用了我们的很多学生担任实习和全职岗位，他评论道，学生非常注重捕捉需求和系统建模。工程部的副总裁评论道，我们的毕业生可与在该公司工作五年的某些软件工程师媲美。他不仅聘用了我们的学生，还资助了一些毕业班项目。

我们的讲师之一 Robert Kuehl 拥有 30 多年的职业发展和消费者与商业影像软件系统的开发管理经验，他对我们的学生所接受的技能培养做出了这样的评价：

总的来说，业界需要：

- *专业精神*作为个人，做事专业，能进行有效的口头和书面交流，且能在不同的团队中高效工作。
- *执行力*作为专业人员，了解如何获取和确定良好的需求，能够把需求转变成实现需求的设计，能有效地编写良好的代码、调试代码和测试代码。他们希望专业人员能够有效地选择和执行软件开发方法论和工具，且管理的项目能够一直按时在预算内交付。
- *技术知识和专长*作为专业人员，熟练掌握当前技术，拥有扎实的计算原理、技术和算法知识，且能够创新。

计算机科学的课程无疑帮助学生掌握了计算的技术知识和专长。软件工程的课程体系也提供了类似的技术基础，但是与其他课程作业进行了结合，以教授专业精神，并获取随之产生的执行力。通过课程项目，这些课程深度覆盖了软件开发生命周期的所有方面。除了项目的技术层面外，这些项目还强调通过实践来学习、团队协作以及交流。

根据我的经验，在需要开发和部署高质量软件的行业中，软件工程的毕业生通常为工作准备得更充分。

Jeffrey Lasky 是信息技术的教授，在 Excellus 公司担任 RIT 的驻场教授。Excellus 是当地的一家 Blue Cross/Blue Shield 健康保险提供商。与 Lasky 的交谈首先向我们揭示了软件工程课程体系带来的一些突出特性。

“RIT/Excellus Blue Cross/Blue Shield 实习计划于 2002 年秋季开始。这个计划由董事、Excellus 架构和集成小组（Excellus Architecture and Integration Group）以及 RIT 的驻场教授共同管理。实习计划向所有主修计算学科的 RIT 本科生开放。

“2004 年，我们指派了一个由六名学生组成的团队（其中计算机科学、信息技术和软件工程各两名学生）为一个优先级高的系统开发项目开发子系统。我们没有刻意地安排团队的组成成员，得来纯属侥幸。学生们很快地意识到，他们各自的技能强项聚集在其学位课程的核心领域周围：程序设计（CS）、数据库和网络（IT）以及设计（SE）。

“虽然所有的课题都让人感兴趣，但是软件工程学生对软件设计模式的使用和解释却最受其他学生的关注，他们很快注意到 SE 的学生在软件系统开发方面的想法不同。特定的模式成为团队对话的一部分，而且往往主导着对话的方向；CS 和 IT 的学生对软件设计中形式化抽象的作用和价值抱有满腔热情。负责指导学生的 Excellus 软件架构师们对于设计模式也有类似的反应。经典的书籍《*设计模式：可复用面向对象软件的基础*》开始出现在他们的案头。此后，又出现在 Excellus 软件开发人员的案头。

Lasky 刻画了本校不同学位的学生的长处，得到了其他机构中的同僚的认同。SE 的学生会提出组件、架构以及组件互动方面的问题，他们偏爱由模型驱动的讨论，因为这种讨论层次较高，更为抽象。CS 和 IT 的学生则倾向于索取工作代码的样例，然后开始自底向上理解系统。CS 学生的编码能力很强，但通常欠缺设计能力，对质量属性的关注也不够。RIT 的软件工程专业对设计与过程的课程进行了平衡，学生熟悉了更多的编程技巧，而有些学生则对大量编码不感兴趣。在对学生的调查中，三分之二的学生更喜欢设计和实现，而其他的学生则对过程更感兴趣（例如，需求、过程改善以及软件质量保证）。

结论

二十年前，当 RIT 开始设立美国第一个软件工程本科专业时，我们就打赌，如果我们设立了这个专业，他们会来——学生和雇主会来。该专业的增长轨迹记录以及超过 90% 的就业率证明，这场赌博卓有成效。把重点放在工程设计、软件产品开发、团队协作和交流上后，可以给在软件开发领域谋职的学生提供量身定制的技能组合。这不仅能让他们成为优秀的初级软件工程师，还能让他们做好充分的准备，在整个职业生涯中蓬勃发展。

queue.acm.org 上的 相关文章

Fun and Games: Multi-Language Development
Andrew M. Phelps and David M. Parks
http://queue.acm.org/detail.cfm?id=971592

Pride and Prejudice:(The Vasa)
George V. Neville-Neil
http://queue.acm.org/detail.cfm?id=1508213

A Conversation with John Hennessy and David Patterson
http://queue.acm.org/detail.cfm?id=1189286

参考资料

1. ABET Computing Accreditation Commission.Criteria for accrediting computing programs; http://www.abet.org/uploadedFiles/Accreditation/Accreditation_Process/Accreditation_Documents/Current/C001%2014-15%20CAC%20Criteria%2010-26-13.pdf.
2. Beck, K., et al.Manifesto for agile software development, 2001; http://www.agilemanifesto.org/.
3. Boehm, B.W., Turner, R.2004.*Balancing Agility and Discipline:A Guide for the Perplexed.* Addison-Wesley, Boston, MA, 2004.
4. Gamma, E., et al.*Design Patterns:Elements of Reusable Object-oriented Software.* Addison-Wesley, Reading, MA, 1995.
5. Holzmann, G.J.*The SPIN Model Checker:Primer and Reference Manual.*Upper Addison-Wesley Educational Publishers, Saddle River, NJ, 2011.
6. Jackson, D. *Software Abstractions:Logic, Language, and Analysis.* MIT Press, Cambridge, MA, 2012.
7. Joint Task Force on Computing Curricula.Computer Science Curricula 2013:Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.
8. Ludi, S., Natarajan, S., Reichlmayr, T. An introductory software engineering course that facilitates active learning.In *Proceedings of SIGCSE Technical Symposium on Computer Science Education*, (2005), 302.
9. Lutz, M.J., Naveda, J.F.The road less traveled:A baccalaureate degree in software engineering. In *Proceedings of the Conference on Software Engineering Education and Training*, (1997).
10. Naveda, F., Lutz, M. Crafting a baccalaureate program in software engineering.In *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education*, (1997).
11. Rochester Institute of Technology, Office of Cooperative Education and Career Services.Salary | Program Overview, 2013; http://www.rit.edu/emcs/oce/students/salary.
12. Rochester Institute of Technology, Department of Software Engineering.Undergraduate curriculum, 2013; http://http://www.se.rit.edu/pagefiles/documents/VSEN%20Flowchart.pdf.
13. Vallino, J. Design patterns—Evolving from passive to active learning.In *Proceedings of Frontiers in Education Conference*, (2003).
14. Vallino, J. If you’re not modeling, you’re just programming: modeling throughout an undergraduate software engineering program.In *Proceedings of the International Conference on Models in Software Engineering*, (2006), 291–300.
15. Vallino, J. What should students learn in their first (and often only) software engineering course?In *Proceedings of the Conference on Software Engineering Education and Training*, (2013), 335–337.

Michael Lutz（mjlvse@rit.edu）从 1976 年开始一直在 RIT 任教。在二十世纪九十年代，他在美国开创并领导了第一个软件工程本科专业的设立工作。他的专业领域包括软件工程教育、软件设计和软件的数学建模。

J. Fernando Naveda（F.Naveda@rit.edu）是 RIT 软件工程系的共同设立者。从 2001 年到 2010 年，他一直担任该系的系主任。他的专业领域包括课程开发以及软件工程方面广泛的学术课题。他在 RIT 已经工作了 21 年，现就职于教务处。

James Vallino（J.Vallino@se.rit.edu）于 1997 年在 RIT 发起设立了软件工程系，此后一直与 RIT 保持联系。2010 年以后，他一直担任该系的系主任。在加入 RIT 之前，他曾先后在 AT&T 贝尔实验室、AVL 公司和西门子公司研究院（Siemens Corporate Research）工作过 16 年。

译文责任编辑：谢涛

DOI:10.1145/2632661.2632664

为瓦解恐怖组织，STONE 算法确定一组关键人物，清除他们后组织的杀伤力可降低到最低程度。

撰稿人：FRANCESCA SPEZZANO、V.S.SUBRAHMANIAN 和 AARON MANNES

重塑恐怖分子网络

2008 年，哥伦比亚革命武装力量 (FARC) 指挥官耐莉·阿维拉·莫雷诺（又名卡丽娜）向哥伦比亚当局自首，此前当局为通缉她而悬赏 100 万美元。恐怖分子（如基地组织的哈立德·谢赫·穆罕默德和库尔德工人党的阿卜杜拉·奥贾兰）抓捕行动通常代价昂贵而颇具风险，卡丽娜的落网则不同，在金钱和人身风险上的成本都极低。我们开发的塑造恐怖组织网络效能（Shaping Terrorist Organization Network Efficiency，STONE）软件平台设计为可识别恐怖分子网络中的一组关键人物，通过各种悬赏计划和抓捕行动清除这些关键人物，最大程度地解除该组织的武装力量。STONE 可以回答谁应当成为悬赏计划的目标。如果政府希望瓦解某一恐怖分子网络并且有资金来清除其中 k 个恐怖分子，则通过该平台可以知道应当将哪 k 个人物设为目标。

这样的清除行动对国际安全是必不可少的。尽管全球各国政府从 2001 至 2008 年在反击恐怖主义上花费了 700 多亿美元，将跨国袭击的数目减少了 34%，但同期恐怖主义死亡人数却每年增长了 67 例。[11] 反恐工作已经将战略举措作为重点，试图通过激励各方以达成一致来解决恐怖主义的根本原因，并且进行了诸多减少恐怖主义的战略性研究来减少恐怖主义。[19] 然而，战略上挫败恐怖组织极为少见，虽然存在一些著名的成功案例（如爱尔兰的临时爱尔兰共和军，以及日本的奥姆真理教）。[4] 因此，目前仍然需要以瓦解恐怖组织网络为目标的战术行动。

STONE 采用了三种新颖的算法：

恐怖分子继任者问题（Terrorist Successor Problem，TSP）。当某个恐怖分子 r 被清除后，算法将推算 r 被另一名恐怖分子 v 替代的可能性；

多恐怖分子继任者问题（Multiple Terrorist Successor Problem，MTSP）。当从一个网络中清除了多名恐怖分子后，算法将推算可能会诞生的新网络及其相关的概率分布；以及，

恐怖分子网络重塑问题（Terrorist Network Reshaping Problem，TNRP）。它使用 MTSP 生成的结果

重要见解

- **确定要清除的恐怖分子，以便最大程度地瓦解其网络，拯救平民和军人的生命，并且节约财务开支。**
- **通过衡量恐怖分子网络的杀伤力，STONE 算法有助于预测拔除恐怖分子头目后谁将成为继任者，准确率可达 80% 以上。**
- **STONE 针对四大真实恐怖组织的相关数据进行了广泛测试：Aaa 基地组织、哈马斯、真主党和虔诚军。**

2009 年 8 月 10 日，哥伦比亚马克思主义 FARC 游击队的前高级成员耐莉·阿维拉·莫雷诺在军人的护送下抵达哥伦比亚麦德林媒体发布会现场，报告她在投降以后所作的和平努力。

RAUL ARBOLEDA/AFP/GETTY IMAGES

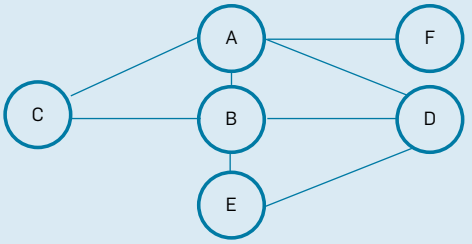
来确定一个需要从网络中清除的恐怖分子集合 k，以此最大程度地降低剩下的网络的预期运作效力。

在我们考虑的某个恐怖分子网络中，每一个顶点（如阿卜杜勒·阿扎姆）可能会有许多属性，其中包括指明他是活着、死亡还是已入狱的状态属性；以及指明他是筹资者、精神领袖还是招募者的角色属性。图 1 概述了一个示例网络，除了顶点属性外，我们还考虑了顶点对西方世界的敌对性、发动袭击的能力，以及抓捕后引起的报复。我们用标记 $\rho(v, p)$ 的属性来表示顶点 v 的属性 p 的值。稍后我们将更加详细地阐述恐怖分子的其他属性。不过，STONE 可以针对任何属性集合运行，而不仅仅是这些属性。网络中的每一个顶点还有一个编码从 1 到 10 的级别，其中 10 为最高；多个人物可能拥有相同的级别。

我们开发了 STONE 算法，并且在军方、警方和我们认识的反恐专家的帮助下进行了测试；所有专家（除一人外）都曾撰写过与反恐相关的著作，也对具体的恐怖主义组织进行过细致分析。该例外是美军的一名退役将军，他在反击恐怖分子和叛乱分子方面有着丰富的经验。我们的测试包含了人工合成图以及开源的真实世界恐怖分子网络数据。我们的专家提供了有关哈马斯 [12]、真主党 [12]、虔诚军 [6,19,20] 和基地组织的数据（包括伊斯兰北非基地组织和阿拉伯半岛基地组织等隶属组织）。[5] 数据中包含恐怖分子的各种属性，以及他们之间的关联。我们的专家也告诉我们网络中已被清除的恐怖分子有哪些、清除的时间为何，以及替代他们的是谁。

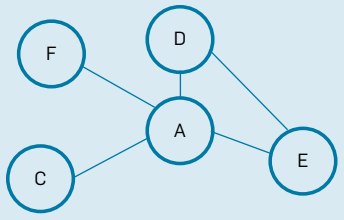
我们对 TSP 求解的准确性进行了测试，如下方所述：我们确定了恐怖分子 r 被清除时可能性最高的继任者 v，以及他 / 她继任 r 的概率 p_{max}，同时也确定了继任 r 的概率“非常接近”（p_{max} 的 2%、3%、4% 或 5% 以内）的所有恐怖分子。我们没有对 MTSP 进行测试，原因有二：它是求

图 1. 示例恐怖组织网络。



人员	级别	角色	容量
A	8	作战部副手	5
B	8	作战部头目	4
C	10	领导人	3
D	7	作战部	4
E	8	作战部副手	3
F	5	作战部	2

图 2. 在移除结点 *B* 后重塑图 1 中的网络。



人员	级别	角色	容量
A	8	作战部头目	5
~~B~~	~~8~~	~~作战部头目~~	~~4~~
C	10	领导人	3
D	7	作战部	4
E	8	作战部副手	3
F	5	作战部	2

图 3. 加权清除页面排名。

Weighted Removal PageRank

$$WRP(v,r) = (1-\delta)\frac{rank(v)}{\sum_{u \in vertices(ON)\setminus\{r\}} rank(u)} + \delta \sum_{u \in nbr(v)\setminus\{r\}} \frac{WRP(u)}{|nbr(u)\setminus\{r\}|}$$

where $rank(v)$ is the rank of v, and $\delta \in [0,1]$ is a number in the $[0,1]$ interval denoting the probability that a vertex's importance is due to network effects. δ is used in the same way as the so-called damping factor in PageRank - and we set its value to 0.85 as in PageRank.

解 TNRP 的中间步骤，而且我们没有相关数据来了解何时有多名恐怖分子在非常接近的时间内被清除。我们使用了对 MTSP 的求解，以及各种杀伤力函数，透过确定网络中的一组 k 个恐怖分子（k 从 1 到 10 不等）来对 TNRP 求解。由于我们无法进行涉及实际恐怖分子清除的实验，我们向专家们展示了结果，请他们对我们建议的有效性按照 1 到 5 分（5 分为最高）进行打分。STONE 的表现不错。我们此处描述的算法也可用于推算与毒品、刑事和洗钱相关的网络。

相关研究

确定网络中的关键人物是通过中心性测量研究的；例如，Memon[13] 使用传统的中心性测量识别关键人物，而 Memon 和 Larson[14] 利用了 Latora 和 Marchiori 的前序工作来定义网络的效率。[9] Memon 和 Larsren[14] 提出了一种地位角色指标来区分此类网络中的捍卫者和追随者，并且定义了“依赖中心性”，即 DC。Memon 和 Larsen[14] 展示了 DC 对 Krebs 的 9/11 网络的价值。[7] Carley[3] 提出了 CONSTRUCT-O 模型，通过整个网络中扩散信息的能力、达成共识的能力，以及该网络执行任务的能力来衡量对该网络的瓦解效果。Carley[2] 采用“孤立”策略，来生成有价值的“目标函数”，可用于此处所述的杀伤力度量。Ozgul[17] 研究了在恐怖分子网络中检测网络基本单元的问题，提出了 GDM 和 OGDM 等算法。Lindelauf[10] 通过博弈论模型研究了恐怖分子网络中通讯需求和藏身需求之间的紧张关系。

相比之下，STONE 开发了谁将替代恐怖分子网络中“被清除”个体的数学模型，其同时考虑了网络结构和顶点属性，以及恐怖分子网络获得运作效力的需求。STONE 利用第一种方法来确定一组顶点被移除后网络将如何演变，并利用该信息来决定要移除哪一组结点；没有事先了解移除的后果就从网络中移除顶点是不明智的做法。

恐怖组织网络

恐怖组织网络 ($\mathcal{ON}$) 是一种包含顶点和边的无方向图形。每个顶点带有一些来自属性集合 VP 绘制中的属性，如前文中所述和图 1 中所示。

我们首先来定义当我们把 $\mathcal{ON}$ 中的顶点 r（被移除的顶点）替换为顶点 v 时会发生什么；我们并不真正用 v 将 r 替代，而只是定义所要发生的情况。假设分析员将图 1 中的顶点 B 替换为顶点 A。STONE 假设 A 保留其所有的关系，并且继承了 B 的关系。一个有趣的问题是 A 的属性将会出现什么情况。在 STONE 中，分析员设定的一个函数来指定从被删除顶点继承的哪些属性，不继承的哪些属性。在我们的示例中，角色和级别属性来自继承，而其他属性则保持不变。图 2 显示了 A 的属性与其之前的属性相同，除了他的级别从过去的 8 级提高到 B 的 9 级。

恐怖分子继任者问题

TSP 解决了恐怖分子网络 $\mathcal{ON}$ 中顶点 r 被移除时谁将替代 r 的结点所发生的情况。STONE 通过三个步骤对 TSP 求解：

顶点属性。除了前文中探讨的属性外，我们定义了特定的网络相关属性：加权清除页面排名 (WRP) 用来衡量顶点的影响，面向属性的聚集系数 (POCC) 用来衡量顶点与邻近顶点的关系紧密度；

候补者替代。我们正式定义了可以替代一个顶点的候补者集合。STONE 允许反恐分析员指定 v 替代被删除的顶点 r 时必须要满足的属性；例如，可能是 v 必须要能够扮演与 r 相同的角色。在图 1 和图 2 中，这意味着如果 D 的角色是募资人而不是执行者，D 将无法替代 B；以及

候补概率。在确定候补者时，我们必须给出给定候补者替代被删除顶点 r 的概率。

顶点的网络属性。此处，我们描述了两个特殊的顶点属性 — WRP 和 POCC — STONE 在使用这两个属性时利用了 Google 的页面排名和社交网络中的聚集系数。STONE 允许使用许多其他典型的中心性测量（如点度中心性和中间中心性），虽然本文中未予阐述。我们选择了 WRP 和 POCC，因为它们分别度量了顶点的影响力，和顶点的连结强度。我们假定影响力和连结性在判断谁最有可能被选为已清除恐怖分子的继任者上发挥重要的作用。

WRP。WRP 用来评定顶点的影响力，具体为通过衡量其周遭的同伴和家人 — 网络中的直接邻居 — 的影响力。拥有高影响力的关系的人员应该也是有影响力的；例如，在基地组织相关的网络中，哈立德·谢赫·穆罕默德 (KSM) 具有很高的影响力，因为他的直接关系中包含基地组织高层领导奥萨马·本拉登和艾曼·扎瓦赫里等。当 KSM 被逮捕（清除）后，与他相关的顶点的 WRP 计算将衡量该顶点的相对影响力。KSM 被阿布·法拉杰·利比替代，而后者根据 WRP 的测量具有很高的影响力分数。WRP 对页面排名 [1] 进行了扩展，来处理三个社交因素：

顶点。$\mathcal{ON}$ 中的顶点具有级别，在计算顶点的重要性时必须予以考虑；

顶点重要性。$WRP(v, r)$ 表示顶点 v 在 $\mathcal{ON}$ 中的重要性，其考虑了顶点 r 即将被清除的事实；以及

无向图。恐怖组织网络是无向图，而不是类似 Web 等的有向图。

v 的 WRP 是一个随着结点 r 的移除而变化的动态量。顶点与将被删除的顶点 r 相关的 WRP 越高，v 就越有可能成为 r 的替代顶点（假设满足稍后阐述的特定条件）。图 3 给出了 WRP 的形式化定义。公式的第一项描述了顶点 v 与关于网络中所有顶点相关的影响力（r 除外），将它乘以 $(1 - \delta)$ 来获得概率排序在顶点重要性中起到的作用。第二项表示的是 v 的关系的重要性，即 v 的每一个邻居 (u) 都将其重要性平均分配给它的邻居，但要被删除的顶点 r 除外。STONE 算法将第二项的和乘以 δ，表示网络影响起到的作用程度为 δ。图 4 概述了图 1 中小型恐怖分子网络的顶点的 WRP，其假设 B 即将被移除。

面向属性的聚集系数。在网络理论中，聚集系数 [21] 用来捕获顶点与网络中其他顶点的关系的“紧密程度”。直观上看，一个恐怖分子与其网络中其他恐怖分子的关系越紧密，他就更有可能获得他们的支持和职位擢升；例如，2003 年 KSM 在巴基斯坦拉瓦尔品第被美军抓获后，他的继任者需要与他的同伙的关系足够紧密，才能提拔到 KSM 在该网络中的高位，而事实上继任 KSM 的阿布·法拉杰·利比就是如此。

从技术上而言，顶点 v 的聚集系数是 v 的邻居之间连接成对的百分比。在 STONE 中，我们引入了面向属性的聚集系数，而不仅仅关注直接的邻居。顶点 v 的 k 近邻包含与 v 的距离不超过 k 的所有顶点。在属于 v 的 k 近邻的顶点中，我们仅关心满足由某分析员指定的属性集合 $\mathcal{P}$ 的所有顶点，因为我们使用面向属性的聚集系数来确定谁有可能替代被删除的 r。因此，我们将注意力限制在具有某些属性的顶点，因为只有具有这些属性的顶点

图 4. 图 1 中小型恐怖分子网络的顶点的 WRP，其假设 *B* 即将被移除。

顶点	WRP	顶点	WRP	顶点	WRP
A	0.36	D	0.24	F	0.12
C	0.14	E	0.14		

图 5. 面向属性的聚集系数。

Property Oriented Clustering Coefficient

$$pocc_{k,\mathcal{P}}(v) = \frac{2 \cdot |\{(u,z) \in edge(ON) \mid u,z \in nbr_{k,\mathcal{P}}(v)\}|}{|nbr_{k,\mathcal{P}}(v)| \cdot (|nbr(v)_{k,\mathcal{P}}| - 1)}$$

where $nbr_{k,\mathcal{P}}(v) = \{u \mid d(u,v) \leq k \wedge (\forall p \in \mathcal{P})\rho(u,p) = 1\}$. As usual, $d(u,v)$ denotes the shortest path distance between u and v. In the case where $|nbr_{k,\mathcal{P}}(v)| \in \{0,1\}$, we set $pocc_{k,\mathcal{P}}(v) = 0$.

图 6. 图 1 中 *C*、*A*、*D*、*E* 和 *F* 替代 *B* 的概率。

v	*P*(*v*, *B*)	*v*	*P*(*v*, *B*)	*v*	*P*(*v*, *B*)
A	0.35	D	0.31	F	0
C	0	E	0.34		

图 7. 多恐怖分子继任者问题和算法梗概。

Multiple Terrorist Successor Problem (MTSP)

Given inputs: (i) a set R of vertices to be removed and (ii) a given $\delta \in [0, 1]$, find a candidate replacement set X_{max} (having some probability p_{max}), as well as all other candidate replacement sets X such that the probability that X replaces R is greater than or equal to $(p_{max} - \delta)$.

MTSP Algorithm Sketch

1. Construct a weighted complete bipartite graph. The first set of vertices S_1 is the set R of vertices to remove. The second set of vertices S_2 consists of all candidates to replace any vertex in R. S_1, S_2 do not overlap. There is an edge from each $r \in S_1$ to each $v \in S_2$ labeled with the logarithm of the probability that v will replace r.
2. Apply an algorithm (such as the Hungarian algorithm[8]) to find a maximum matching over this graph. Return this result.

图 8. 杀伤力函数。

Lethality Functions

$$L_1(ON) = \sum_{v \in VPR(ON)} rank(v) - \sum_{v' \in VA(ON)} rank(v')$$

$$L_2(ON) = \sum_{v \in VPR(ON)} deg(v) \cdot rank(v) + - \sum_{v' \in VA(ON)} deg(v') \cdot rank(v')$$

$$L_3(ON) = \sum_{v \in VPR(ON)} WRP(v, \emptyset) \cdot rank(v) + - \sum_{v' \in VA(ON)} WRP(v, \emptyset) \cdot rank(v')$$

才有可能影响到替代顶点的甄选；例如，虔诚军指挥官的甄选 — 如果当前的指挥官扎奇尔·勒赫曼·拉赫维被清除 — 可能仅由虔诚军当前的头目、虔诚军作战部门的某些高级成员、虔诚军长老议会成员或顾问委员会决定。其他与扎奇尔·勒赫曼·拉赫维直接关联的人以及他的 k 近邻可能不会对他继任者的甄选产生真正的影响。[a] WRP 反映的是单个恐怖分子的影响力，POCC 捕获的是该恐怖分子与其同伙在组织中合作的紧密程度；POCC 在图 5 中定义。[b]

候补者替代。 STONE 假设分析员指定了各个顶点属性的权重来定义各个属性的相对重要性；例如，顶点的敌对性权重可能被设为 0.4，而顶点的 POCC 权重可能被设为 0.2，这表示根据分析员对该组织的情况的评估，POCC 的重要性仅仅是顶点的敌对性的一半。

假设 α_i 是与我们这一套属性中各个属性 p_i 相关的权重，α 的总和为 1。[c] 每个可能会替代已删除顶点 r 的顶点 v 具有一个替代值 $rv(v, r)$，定义如下

$$rv(v, r) = \Sigma_{p^i \in VP} \alpha_i \cdot \rho(v, p_i)$$

STONE 允许分析员指定任意一组属性 C，顶点需要具有这些属性才能成为将被删除的顶点 r 的合适的替代结点；例如，分析员可以指定顶点 v 必须是恐怖分子网络中的领导层成员，或者与将被删除的顶点 r 所属同一部门（作战部）的成员。STONE 支持此处未涵盖的许多网络中心性属性（如点度中心性、中间中心性和接近中心性等）。我们现在定义顶点 v 被视为顶点 r 的候补替代的时间。

v 何时成为替代 r 的候补者呢？当 v 满足条件 C、$\text{rank}(v) \leq \text{rank}(r)$，并且不存在符合以下条件的其他顶点 u（除 v 和 r 外）：u 满足前面所述两个条件，并且 u 的替代值明显大于 v 的该值。

示例 1。 考虑图 1 中的网络，其中 B 为要删除的顶点，并且假设 C 规定 B 和 v 之间的距离必须为 1。此外，假设 $rv(v, B)$ 已通过计算获得，并且考虑了 WRP ($\alpha_0 = 0.2$)、POCC ($k = 2$) 和 $P = \emptyset$ ($\alpha_1 = 0.2$)，以及结点级别 ($\alpha_2 = 0.4$) 和结点容量地位 ($\alpha_3 = 0.2$)。[c] 那么，可能会替代顶点 $r = B$ 的候补者为：

A，$rv(A, B) = 0.672$

D，$rv(D, B) = 0.588$

E，$rv(E, B) = 0.634$

a 尽管拉克维因为其在 2008 年印度孟买的攻击而在巴基斯坦坐牢，但有新闻来源报道他在监狱中指挥行动。

b 分子是与 v 的距离为 k 个单位以内、与之直接相连并且具有 P 中属性的顶点对的数量；分母是 v 的邻居（具有 P 中的所有属性、并且与 v 的距离为 k 个单位）的总数。该比值提供所需的数量。

c 级别和容量的值随着它们的最大值缩放。

此处，A 是最佳候补者，E 其次。

候补者概率。我们现在定义候补者的概率，即 v 将替代 r 的概率：

$$\mathbf{P}(v,r)=\frac{rv(v,r)}{(\Sigma_{u\ is\ a\ candidate}\ rv(u,r))}$$

替代被删除的结点 r 的概率是，v 的替代值除以所有可能替代 v 的候补者的替代值之和；例如，图 6 中列出了图 1 中替代 B 的 C、A、D、E 和 F 的概率。我们可以观察到，$\mathbf{P}(C,B)=0$，因为 C 的级别大于 B，而 $\mathbf{P}(F,B)=0$，因为 B 和 F 的距离大于 1。

多恐怖分子继任者问题

如果分析者不打算删除单个顶点 r，而是要移除一组顶点 R 呢？例如，2012 年 6 月在两次紧接着的行动中抓捕了两名虔诚军恐怖分子。据报道，曾经现身巴基斯坦控制室指挥 2008 年 11 月孟买袭击的阿卜·金德尔在印度、沙特和美国的一次联合行动中被抓获；而虔诚军最高爆破专家顿达则在 2013 年 8 月于印度尼泊尔边界被抓获。大多数安全机构都以同时抓捕多名恐怖分子为目标。在这样的情形中，可能会存在许多网络（例如，如果 n 个人可能替代 阿卜·金德尔，而 m 个人可能替代顿达，那么就可能会出现 mn 个新网络）。MTSP 的设计目标是帮助确定新网络的概率。

现在我们假设，像单顶点被移除的情形中一样，分析员指定了替代顶点必须要满足的条件 C。要移除的顶点集合 R 的候补者替代者集合 X_R（并且满足条件 C）是一个对集合（r, c_r）（r 是要删除的顶点，c_r 则是其替代顶点），它们满足以下四个条件：

*(i) 每个被删除的顶点具有一个替代顶点。*对于每个 $r\in R$，有一个对 $(r,c_r)\in X_R$；并且

(ii) 替代顶点无法被移除。 $\{c_r \mid (r,c_r)\in X_R\}\subseteq(V-R)$；并且

*(iii) 替代结点必须为候补者。*对于每个对 $(r,c_r)\in X_R$，c_r 是与条件 C 相关的替代 r 的候补者；并且

*(iv) 候补者只能替代一个顶点。*不会有两个 $r,r'\in R$，(r,c_r) 和 (r',c_r) 都在 X_R 中。

以下示例返回到我们运行中的例子，演示分析员移除图 1 和示例 1 中的顶点集合 $\{B,C\}$ 时会发生什么。

*示例 2。*假设分析员要移除集合 $R=\{B,C\}$。B 的候补替代者集合为 $\{A,D,E\}$，而 C 的候补替代者集合为 $\{A,B\}$。A 无法同时替代 B 和 C，而 B 则不是替代 C 的有效候补者，因为 B 要被移除。R 的替代者集合为 $X'_R=\{(B,D),(C,A)\}$，而且 $X''_R=\{(B,E),(C,A)\}$。

可能会有许多候补替代者集合 X_R 与 R 和 C 相关。X_R 将真正替代 R 的概率 $\mathbf{P}(X_R)$ 仅仅是 c_r 将替代顶点 r 的各个概率 $\mathbb{P}(c_r,r)$ 的乘积。要了解其中的原理，可返回到图 1 中的示例和示例 2。

*示例 3。*假设 $\mathbb{P}(B,C)=0$，$\mathbb{P}(A,C)=1$，$\mathbb{P}(D,B)=0.31$，并且 $\mathbb{P}(E,B)=0.34$。所以，替代者集合 $X'_R=\{D,A\}$ 的概率等于 $(0.31\cdot1)/(0.31+0.34)=0.46$。而替代者集合 $X''_R=\{E,A\}$ 的概率等于 $(0.34\cdot1)/(0.31+0.34)=0.54$。

当给定了要删除的顶点集合 R 时，假设 X_{max} 为概率最大的候补替代者集合。由于参数设置中存在许多不确定性（例如，各个属性的权重选择，面向属性的聚集系数定义中 k 的选择，以及反恐分析员设置的条件 C 的选择），STONE 算法不希望仅仅把 X_{max} 返回给分析员。相反，STONE 返回概率最多比 X_{max} 的概率小 δ 个百分点的所有候补替代者集合 X_R（见图 7）。注意，X_{max} 不是此问题的输入；STONE 算法自动求解，而不将 X_{max} 作为输入。STONE 开发人员证明，MTSP 可以通过利用 Murty[16] 的有效算法（以成本升序方式枚举加权双向完全图中最小匹配的全部答案）在多项式时间内求解，如图 7 中所示。通过利用 Murty，[16] STONE 算法以概率降序的方式枚举最大匹配的所有答案，直到 STONE 获得答案 X_R，以便 $\mathbb{P}(X_{max})-\mathbb{P}(X_R)>\delta$。

杀伤力函数

当分析员推荐从网络中移除的结点集合时（例如通过抓捕行动），分析员必须考虑可能会产生的新网络，及其杀伤力。我们之前已探讨过前者，现在探讨如何衡量网络杀

图 9. 恐怖分子网络时间序列。

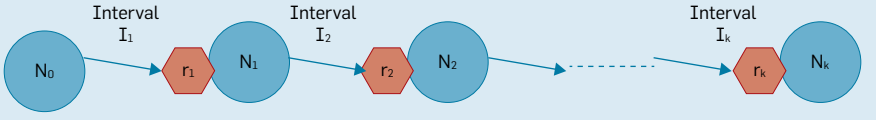
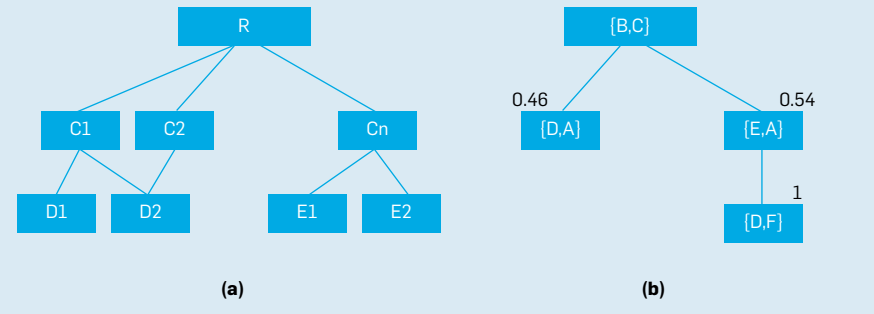


图 10. 替代关系图。



伤力的方式，其中包括四种方法：L_1 评估推崇暴力和反对暴力的结点的级别差异；L_2 通过相关顶点的度数来调节 L_1；L_3 利用 WRP 而不是节点度数来调节 L_1；以及 L_4 通过回归将 L_1、L_2 和 L_3 与相关攻击关联。

如果顶点的能力和敌对性都超过分析员定义的阈值，则该顶点是推崇暴力的；否则，该顶点属于反对暴力。STONE 使用 $VPR(\mathcal{ON})$ 和 $VA(\mathcal{ON})$ 分别表示 $\mathcal{ON}$ 中所有推崇暴力和反对暴力的顶点集合；图 8 包含了三个杀伤力函数。

L_1 表明网络的杀伤力等于推崇暴力顶点的级别之和减去反对暴力的顶点的权重之和；也就是说，推崇暴力的个体的暴力受到了反对暴力的个体的缓和；例如，二十世纪九十年代和本世纪初形成的恐怖组织印度伊斯兰教学生运动具有一个推崇暴力的派别（后来分离出去，成为印度圣战者组织）和反对暴力的派别（延续到现在）。

示例 4。思考图 1 中的网络 $\mathcal{ON}$，并假设顶点 A、B 和 D 是推崇暴力的，而 C、E 和 F 是反对暴力的。那么，$L_1(\mathcal{ON}) = wt(A) + wt(B) + wt(D) - wt(C) - wt(E) - wt(F) = 1$。

L_2 指出，顶点的杀伤力是顶点的度数乘以该顶点的级别，而网络的杀伤力等于推崇暴力的顶点的杀伤力得分之和减去反对暴力的顶点的杀伤力之和。第三个杀伤力函数类似，但使用的是 WRP 而不是度数。

示例 5。继续阐述示例 4，分析员得到 $L_2(\mathcal{ON}) = 48$，而 $L_3(\mathcal{ON}) = 2.69$。

尽管可能还有许多其他杀伤力函数，但此处我们不一一阐述，这里我们介绍一个混合杀伤力函数：

混合杀伤力函数。根据一位评论员和反恐专家 Daveed Gartenstein-Ross（来自基于华盛顿特区的智库保卫民主基金会）独立提出的建议，以攻击数据为基础来定义杀伤力。因此，我们尝试将网络结构与攻击数据关联。如图 9 中所述，与特定组织相关的网络在最初一段时期 I_1 内是网络 $\mathcal{ON}_0$，直到恐怖分子 r_1 被清除，形成网络 $\mathcal{ON}_1$。在一段时期 I_2 后，恐怖分子 r_2 被清除，形成网络 $\mathcal{ON}_2$。在每一个时间段 I_j 内，该组织发动了一定数量 A_j 的攻击。变量 A_j 是网络 N_j–1 的杀伤力的衡量指标。对于每个组织，我们据此能够建立一张表格。此表格第 j 行对应于时期 I_j，列中包含的 $L_1(\mathcal{ON}_j)$、$L_2(\mathcal{ON}_j)$ 和 $L_3(\mathcal{ON}_j)$ 值是自变量，而 A_j 则是因变量。通过标准的多元回归分析，[15] 我们了解到混合杀伤力函数 L_h 将 L_1、L_2 和 L_3 返回的值与各个杀伤力变量关联。STONE 开发者将这称为混合函数 h。我们通过实验来检查该混合函数在预测攻击次数上的有效性。我们利用我们的虔诚军网络数据集 (1990–2012) 和基地组织数据集 (2001–2013) 进行了测试。在我们虔诚军数据集的情形中，STONE 发现我们获得的混合杀伤力函数 L_h 可以帮助我们预测虔诚军的攻击。实际攻击次数和基于隐蔽数据预测的虔诚军攻击次数之间的皮尔森关联系数为 0.83。然而，当我们从较小的基地组织数据集获得 L_h 时，基地组织实际攻击次数和我们预测值之间的皮尔森关联系数为 0.652。

图 11. 可能的恐怖分子网络。

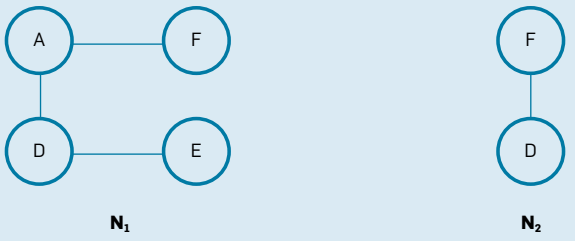


图 12. 网络重塑算法。

Algorithm 1 STONE-Reshape network reshaping algorithm

Input: Organizational network ON, maximum set size k, condition C, set EX of unremovable vertices.
Output: Set R of nodes to remove.
function RESHAPE(ON, k)
  $R = \emptyset$ % nobody to remove yet
  Set ℓ to the lethality of ON
  $continue = true$
  do
    Let r be a vertex in $ON \setminus (R \cup EX)$ s.t. r satisfies C and $L_{EV}(ON, R \cup \{r\})$ is minimized
    Let ℓ_r be the expected lethality of the network if $R \cup \{r\}$ is removed.
    If($\ell_r \leq \ell$) % i.e. if removing r leads to further weakening of the network
      $R = R \cup \{r\}$% then remove r
      $\ell = \ell_r$ % updated exp. lethality of the network
    Else $continue = false$
    EndIf
  while($continue \wedge |R| \leq k$)
  return R
end function

L_h 因此能够为预测恐怖组织的攻击次数提供准确的方法，其准确性随着分析员拥有的数据量而提高。

要移除哪些顶点?

为了决定要从网络中移除哪些 k 个顶点（可能要排除某些顶点集合 EX），STONE 首先定义可能的恐怖分子网络 (PTN) 以及替代关系图解（图形）。图 10a 列出了一个替代关系图解示例。假设 R 是考虑要移除的顶点集合；R 是替代关系图的根。我们知道有几个候补者可能在等待替代 R 中的顶点。我们使用之前的定义，把这些候补者称为 $C_1,\cdots,C_n$。每个 C_i 是 R 的子结点，用概率 p_i 标记其关联，其中 p_i 是 C_i 替代 R 的概率。如果 C_i 会替代 R，STONE 需要找出顶点集合 C_i 的替代者。图 10a 通过示例概述了这一关系，其中 C_1 拥有两个可能的候补替代者 —D_1 和 D_2。类似地，C_2 是要替代 R 的候补者集合，但只有一个候补者来替代它 —D_2；请参见图 10b 了解所述示例的替代关系图。

一个可能替代链 (PRC) 是从替代关系图的根结点 R 到任何叶子结点的路径。这一路径上的结点标记代表移除的顺序，以及 PRC 中的替代者；例如，路径 R, C_2, D_2 代表从网络中删除 R，其替代者为 C_2，而 C_2 的替代者为 D_2。每个 PRC $X_1,\cdots,X_m$ 生成一个 PTN，如下方所示：从恐怖分子网络 $\mathcal{ON}$ 开始，假设 $\mathcal{ON}_1$ 是通过将网络 $\mathcal{ON}$ 中的 R 替换为 X_1 而获得的网络。类似地，假设 $\mathcal{ON}_2$ 是将网络 $\mathcal{ON}_1$ 中 X_1 替换为 X_2 而得到的结果。STONE 重复这一过程，直到网络 $\mathcal{ON}_{m-1}$ 中的 X_m−1 被替换为 X_m。生成的网络是可能的恐怖分子网络。

这一示例中包含两个 PTN —N_1 和 N_2（见图 11）。与某一路径相关的 PTN 的概率是标记这些边的概率的乘积。PTN N_2 的概率 $\mathbb{P}(N_2)$ 是 $0.54 \cdot 1 = 0.54$。

替代关系图中的叶结点可通过多条路径到达，因为同一 PTN 可能会由多个 PRC 生成。该 PTN 的概率是允许 STONE 到达该叶结点的每个替代链相关的 PTN 概率之和。到达该叶结点在图 10a 中就是与叶结点 D_2 对应的 PTN。

STONE 解决了在恐怖组织网络中识别 k 个恐怖分子的集合的问题，抓捕和清除这些恐怖分子能够使生成的网络的杀伤力降到最低。

假设 R 是要移除的顶点集合，$\text{PTN}(\mathcal{ON}, R)$ 是可能生成的所有 PTN 的集合，而 L 则是杀伤力函数。当 R 被移除时，网络的统计学预期杀伤力被 STONE 定义为：

$$L_{EV}(\mathcal{ON},R) = \Sigma_{N\in PTN(\mathcal{ON},R)}\, \mathbb{P}(N)\cdot L(N)$$

回到我们所述的示例，可能网络 N_1 的杀伤力 L_2 为 $L_2(N_1) = 17$，而 $L_2(N_2)$ 则为 2。那么，预期杀伤力就是 $\text{LEV}(\mathcal{ON},R) = \Sigma_{N\in\text{PTN}(\mathcal{ON},R)}\, \mathbb{P}(N)\cdot L_2(N) = (0.46\cdot 17) + (0.54\cdot 2) = 8.9$。

TNRP。

输入。恐怖组织网络 $\mathcal{ON}$、每个恐怖分子被清除时必须满足的条件 C，以及数字 k。

输出。顶点集合 R，其中

(i) $|R| \le k$ 并且

(ii) 每个 $r \in R$ 满足条件 C，而且

(iii) 不存在满足 (i) 和 (ii) 并且 $L_{EV}(\mathcal{ON}, R') < L_{EV}(\mathcal{ON},R)$ 的顶点集合 R'。

分析员可以指定被清除恐怖分子必须满足的条件 C；至少，该顶点必须处于*活着*或*其他*状态。我们也可以想象，分析员表示某些恐怖分子由于潜在的政治后果而不应当被清除；例如，尽管美国国务院于 2012 年宣布提供 1000 万美元赏金以获得可起诉虔诚军领导人哈菲兹·赛义德的信息，但他依然可以在巴基斯坦境内自由活动，举办记者会并在公开集会中演讲。

由于求解 TNRP 几乎是一个 NP 难问题，STONE 开发者开发了一种贪心算法以进行求解，如下方所示：算法 1（图 12）以 $R = \emptyset$ 开始。在每一迭代中，它寻找可删除顶点 r，它将最大程度地减弱预期

杀伤力（图 12 中算法的第 8 行）。第 10 行检查是否要将 r 添加到 R 来减弱预期杀伤力。若为是，则 r 添加到 R 中（第 11 行），STONE 算法则将重复，直到 R 拥有 k 个元素，或者添加更多元素到 r 不会减弱预期的杀伤力。

在我们包含 L_3 和 $k = 2$ 的示例中，算法 1 建议删除 D 和 F，将网络的预计杀伤力降低到 -8.87。

实施和实验

图 13. 实验参数设置。

设置	α_{WPR}	α_{POCC}	α_{rank}	α_{host}	α_{comp}	α_{BB}
1	0.1	0.1	0.6	0.066	0.067	0.067
2	0.8	0	0	0.066	0.067	0.067
3	0.267	0.266	0.267	0.066	0.067	0.067
4	0.2	0.2	0.4	0.066	0.067	0.067
5	0.4	0.2	0.2	0.066	0.067	0.067
6	0.4	0.4	0	0.066	0.067	0.067

图 14. 实验 1 结果。

设置	数据集	δ = 2%	δ = 3%	δ = 4%	δ = 5%
1	示例	0.905(2)	0.925(3)	0.95(3)	0.99(5)
	基地组织	1(1.81)	1(1.90)	1(2)	1(2)
	虔诚军	0.667(4)	0.667(6)	0.833(7)	0.833(7)
	哈马斯	0.8(3.6)	0.8(4)	0.8(4.4)	0.8(4.4)
	真主党	0.8(4.2)	0.8(5.6)	0.8(8.2)	0.8(10.2)
2	示例	0.56(2)	0.56(2)	0.605(2)	0.635(2)
	基地组织	0.90(1.27)	1(1.54)	1(1.54)	1(1.54)
	虔诚军	0.5(5)	0.667(7)	0.833(7)	0.833(8)
	哈马斯	0.8(2)	0.8(2.8)	0.8(3.4)	0.8(3.8)
	真主党	0.4(4.2)	0.6(8)	0.6(8.8)	0.6(9)
3	示例	0.53(2)	0.80(3)	0.865(4)	0.935(5)
	基地组织	0.90(1.45)	0.90(1.72)	0.90(1.72)	0.90(1.72)
	虔诚军	0.833(5)	0.833(7)	0.833(7)	0.833(8)
	哈马斯	0.4(1.6)	0.8(2.4)	0.8(3.2)	0.8(4.2)
	真主党	0.8(3.8)	0.8(4.8)	0.8(7)	0.8(10)
4	示例	0.835(3)	0.93(4)	0.97(4)	0.985(5)
	基地组织	0.90(1.63)	0.90(1.72)	1(1.90)	1(2)
	虔诚军	0.833(5)	0.833(7)	0.833(7)	0.833(8)
	哈马斯	0.8(2.4)	0.8(3.8)	0.8(4.4)	0.8(4.8)
	真主党	0.8(4.2)	0.8(5.4)	0.8(7)	0.8(10.2)
5	示例	0.765(3)	0.865(4)	0.92(5)	0.94(5)
	基地组织	0.90(1.45)	0.90(1.54)	0.90(1.72)	1(1.81)
	虔诚军	0.833(6)	0.833(7)	0.833(7)	0.833(8)
	哈马斯	0.6(1.8)	0.8(2.8)	0.8(3.8)	0.8(4.6)
	真主党	0.8(3.2)	0.8(5.6)	0.8(8)	0.8(10.8)
6	示例	0.16(2)	0.26(3)	0.32(3)	0.37(3)
	基地组织	0.72(1.27)	0.72(1.36)	0.72(1.36)	0.72(1.36)
	虔诚军	0.667(4)	0.833(6)	0.833(7)	0.833(8)
	哈马斯	0.4(2)	0.4(2.2)	0.4(2.4)	0.4(2.4)
	真主党	0.6(2.6)	0.8(3.6)	0.8(5.6)	0.8(10.6)

图 15. 实验 2 结果。

设置	基地组织	哈马斯	真主党	虔诚军
1	5	4	4.6	3.5
2	5	4	4.6	4
3	5	4	4.6	4
4	5	4	4.6	4
5	5	4	4.6	4
6	5	4	4.6	4

我们以 1,500 行 Java 代码实施了本文中所述的所有算法，并在一台运行 Linux 操作系统的 AMD Opteron Quad-Core 2354（2.2GHz、8GB RAM）的计算机上运行这些代码。我们的实验数据集包含 50 个网络；顶点的预计继任者由一部权威反恐书籍的作者和一名美军退役将军挑选。我们也使用了四个真实数据集：(i) 基地组织（101 个结点，121 条边），由基地组织专家通过扩展 Sageman [18] 的数据构建而成；(ii) 哈马斯（78 个结点，139 条边），由其中一名作者、哈马斯分析专家构建而成；(iii) 真主党（60 个结点，64 条边），由外部的真主党专家构建而成（该专家在黎巴嫩进行了广泛的研究，并从阿拉伯语、英语和法语来源构建该数据集）；以及 (iv) 虔诚军（153 个结点，220 条边），由两名作者在撰写虔诚军相关书籍《*Computational Analysis of Terrorist Groups:Lashkar-e-Taiba*》（恐怖组织分析：虔诚军）时构建。所有这些数据集都代表了真实的实际情况。由于 STONE 不需要训练，所有数据都直接用于测试。

实验 1。 我们测试了哪些设置可以获得最高的 TSP 预测准确性，并让 δ 等于 2%、3%、4% 或 5%；图 13 列出了我们使用的六项设置。常量 α_{WRP} 和 α_{POCC} 表示 WRP 的权重；POCC 表示 WRP 和 POCC 的权重；α_{rank} 表示顶点的级别；α_{host} 表示顶点对西方世界的敌对性；α_{comp} 表示顶点从事恐怖袭击的能力；而 α_{BB} 则表示报复性。图 14 括号中的数值是 STONE 算法返回的结果的平均值。STONE 的目标是以最高的准确性预测被清除恐怖分子的继任者，同时尽可能向分析员提供最少数量的候补继承者。设置 3 和 4 是最佳的，返回最少的候补者并且准确度在 80% 以上，即使 δ = 2% 时。从设置 3 和 4 看来，候补者的级别似乎是最重要的因素，并列在其后的则是其影响力 (WPR) 和网络强度 (POCC)。

实验 2。由于不可能通过清除真正的恐怖分子来衡量 STONE-Reshape 算法的准确性，而且也没有此类清除行为的真实历史数据，我们通过将预测结果与反恐专家的观点进行比较来测试算法的准确性，反恐专家对结果从 1 到 5 分（5 分表示最佳）进行打分。我们对基地组织、哈马斯、真主党和虔诚军数据集测试了所有三个杀伤力函数，以及所有六项设置；图 15 包含专家们在考虑了杀伤力函数 L_3 和所有六项设置的满意度水平。

结论

STONE 解决了在恐怖组织网络中识别 k 恐怖分子集合的问题，抓捕和清除这些恐怖分子能够使生成的网络的杀伤力降到最低。为达到这样的性能，STONE 解决了三个问题：恐怖分子继任者、多恐怖分子继任者，以及恐怖分子网络重塑。我们首先开发了一种预测谁将替代“被清除”恐怖分子的方法。通过四个真实世界恐怖分子网络数据集，我们演示了在 80% 以上的剿灭恐怖分子案例中，实际替代被清除恐怖分子的人确定为属于 STONE 预测的最有可能替代的 2% 概率范围内的人；此外，这 2% 的顶级候补者中基本上仅包含几个恐怖分子（图 13 中的设置 3 和 4）。我们演示了如何推断删除一组恐怖分子而可能产生的新网络的概率分布。我们使用了步骤一和二开发 STONE-Reshape 算法来确定包含 k 或更少个人的集合，从网络中清除这些人后可最大程度地降低其运作效能。STONE 允许分析员针对可被清除的人（例如，分析员可以确定某些人不可被清除）和应该考量的顶点属性（分析员可以对不同的组织使用不同的属性）设置限制，也可以设置谁可以替代谁的限制。

本文工作是研究从恐怖分子网络中清除谁的这一重要问题的开端。当某一顶点被移除时，可能会出现权力斗争，导致组织分裂。因此，我们能否调整 STONE-Reshape 算法来预测何时会出现这样的分裂？类似地，级别较高的恐怖分子可能会偶尔“下放”以填补级别较低的恐怖分子被清除后出现的空缺。或者，空缺的职位可能被分配给级别较高的人。我们在此处没有考虑这一可能性，虽然它应当予以解决。一个组织可能会有多个隶属组织（例如，基地组织中有伊斯兰北非基地组织和阿拉伯半岛基地组织）。因此，在确定如何瓦解某一组织时，必须探索组织和隶属组织之间的关系。我们不希望瓦解组织的方式实际上会导致其中一个杀伤力较强的隶属组织变得更为强大。因此，人们有充足的机会进行进一步的研究。

作为总结，我们重申这样一个事实，挫败恐怖分子行动的战术方法可以解决恐怖组织的一组诉求的“症状”（攻击）。在可以这样做的情形中，可以尝试奖赏方法；例如，卢旺达政府的裁军、遣散和重整计划拔除了曾在 1994 发动卢旺达种族大屠杀的胡图族军事武装的大部分力量。然而，这样的行动并不总会成功。反恐任务的战术性行动必须在无辜民众存在被攻击风险的时候进行，而战术方法和战略奖赏计划的整合在与恐怖组织斗争时应当予以审慎的考量。

鸣谢

本文基于“STONE:Shaping Terrorist Organizational Network Efficiency”（STONE：塑造恐怖组织网络效能）这篇论文，其发表于 ***2013 IEEE/ACM ASONAM***（*社交网络分析与挖掘的进展*）*大会会刊*（加拿大尼亚加拉大瀑布，8 月 25–28 日）。ACM Press, New York, 2013. 部分研究资金来自美军研究办公室（授权编号 W911NF0910206）。

参考资料

1. Brin, S. and Page, L. The anatomy of a large-scale hypertextual Web search engine.*Computer Networks 30*, 1–7 (Apr. 1998), 107–117.
2. Carley, K.M.Destabilization of covert networks. *Computational & Mathematical Organization Theory 12*, 1 (Apr. 2006), 51–66.
3. Carley, K.M., Lee, J.-S., and Krackhardt, D. Destabilizing networks.*Connections 24*, 3 (2002), 79–92.
4. Cronin, A.K.*How Terrorism Ends:Understanding the Decline and Demise of Terrorist Campaigns*.Princeton University Press, Princeton, NJ, 2010.
5. Gartenstein-Ross, D. *Bin Laden's Legacy:Why We're Still Losing the War on Terror.* John Wiley & Sons, Inc., New York, 2011.
6. John, W. *Caliphate's Soldiers:The Lashkar-e-Tayyeba's Long War.*Amaryllis and the Observer Researcher Foundation, New Delhi, India, 2011.
7. Krebs, V.E.Mapping networks of terrorist cells. *Connections 24*, 3 (2002), 43–52.
8. Kuhn, H.W.The Hungarian method for the assignment problem.*Naval Research Logistics Quarterly 2*, 1–2 (Mar. 1955), 83–97.
9. Latora, V. and Marchiori, M. How the science of complex networks can help developing strategies against terrorism.*Chaos, Solitons & Fractals 20*, 1 (Apr. 2004), 69–75.
10. Lindelauf, R., Borm, P., and Hamers, H. The influence of secrecy on the communication structure of covert networks.*Social Networks 31*, 2 (May 2009), 126–137.
11. Lomborg, B. Is counterterrorism good value for money?*NATO Review Magazine* (Apr. 2008).
12. Mannes, A. *Profiles in Terror:A Guide to Middle East Terror Organizations*. Rowman and Littlefield Publishers, Lanham, MD, 2004.
13. Memon, B.R.Identifying important nodes in weighted covert networks using generalized centrality measures.In *Proceedings of the European Intelligence and Security Informatics Conference* (Odense, Denmark, Aug. 22–24).IEEE Computer Society, 2012, 131–140.
14. Memon, N. and Larsen, H.L.Practical algorithms for destabilizing terrorist networks.In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics* (San Diego, CA, May 23–24). Springer, 2006, 389–400.
15. Mendenhall, W. and Sincich, T. *A Second Course in Statistics:Regression Analysis*.Pearson, Upper Saddle River, NJ, 2011.
16. Murty, K.G.An algorithm for ranking all the assignments in order of increasing cost.*Operations Research 16*, 3 (May–June 1968), 682–687.
17. Ozgul, F., Bondy, J., and Aksoy, H. Mining for offender group detection and story of a police operation.In *Proceedings of the Sixth Australasian Conference on Data Mining and Analytics* (Gold Coast, Australia, Dec. 3–4).Australian Computer Society, Sydney, 2007, 189–193.
18. Sageman, M. *Understanding Terror Networks*. University of Pennsylvania Press, Philadelphia, 2004.
19. Subrahmanian, V., Mannes, A., Sliva, A., Shakarian, J., and Dickerson, J. *Computational Analysis of Terrorist Groups:Lashkar-e-Taiba.* SpringerLink :Bücher.Springer London, Ltd., 2012.
20. Tankel, S. *Storming the World Stage:The Story of Lashkar-e-Taiba.* C. Hurst & Co.(Publishers) Ltd., London, U.K., 2011.
21. Watts, D.J. and Strogatz, S.H.Collective dynamics of ‘small-world’ networks.*Nature 393*, 6684 (June 4, 1998), 440–442.

Francesca Spezzano (spezzano@umiacs.umd.edu) 是马里兰州大学公园市马里兰大学高级计算机研究所的教学研究助理。

V.S. Subrahmanian (vs@umiacs.umd.edu) 是马里兰大学计算机科学教授，以及数字国际政府中心主任。

Aaron Mannes (amannes@umd.edu) 是马里兰大学高级计算机研究所的教学研究助理，以及公共政策学院博士研究生。

译文责任编辑：唐杰

DOI:10.1145/2628036

虽然最大流算法历史悠久，但突破性的进展仍然层出不穷。

ANDREW V. GOLDBERG， ROBERT E. TARJAN

高效的最大流算法

最大流问题及其对偶最小割问题是经典的组合优化问题，在科学和工程领域应用广泛；例如，Ahuja 等人对此进行过讨论。[1] 该问题是线性规划中一个特例，可以使用普通的线性规划技术或经过特殊化处理的线性规划技术求解（比如网络单纯形法[9]）。不过，专用算法的效率更高。不仅如此，为计算最大流而开发的算法设计技术和数据结构也可用于解决其他问题。虽然最大流问题是线性规划中的一个特别的场景，但它的通用性很强，所以诸多重要的问题（比如二分图最大匹配问题）均可归结为最大流问题。

在本文中，我们概述了高效最大流算法背后的基本技术，我们从基础的最大流算法的历史及其背后的基本理念开始。然后我们详细地探讨了各种算法。我们把自己的研究限制在基本的最大流算法范围内，没有涉及有趣的特殊场景（比如无向图、平面图和二分图匹配），也没有进行泛化（比如最小费用和多商品流问题）。

在正式定义最大流和最小割问题前，我们先来看看与其有关的两个简单示例：在最大流的示例中，假设我们用一张图来表现从油井到油库的输油管网。图中每一条弧均有容量限制，即该弧对应的管道中流量（每秒升数）的最大值。我们的目标是找出从油井到油库的最大运输流量（每秒升数的最大值）。在最小割问题的示例中，我们希望找出总容量最小的管道集，移除这些管道后，会切断油井和油库之间的输送管线（最小割）。

最大流最小割定理指出，最大流的值与最小割的容量相等。该基本定理的应用领域非常广泛，特别是在最大流算法的设计中应用颇多。

我们按多项式（时间）算法和强多项式（时间）算法对网络流算法进行区分。在输入网络中，我们把顶点和弧的数量分别记为 n 和 m。对于多项式算法，弧上的容量为整数，其中 U 指最大容量；容量可用 $O(\log U)$ 位整数表示。（在实践中，假设容量为整数较为合理；因为在计算机硬件的实现中，即使浮点数也用整数表示）。多项式（时间）算法是指在最坏情况下算法运行时间不超过 $n, m, \log U$ 的多项式。对于很多应用而言，算法会以机器中的字为单位对数据进行处理，且基本的算术运算只需要单位时间。我们假定，在阐述多项式边界时，情况跟上述情况相同。强多项式（时

重要见解

- **沿最短路径增广的理念引出了多项式时间算法。**
- **数据结构和细粒度的操作让算法变得更快。**
- **在指定弧的长度时，如果依据剩余容量进行区分，则能提升时间边界。**

插图由 JURGEN ZIEWE 提供

间）算法是一种在最坏情况下其运行时间仍不超过 n, m 的多项式的算法，即使边上的容量是任意的实数，这里假设在我们的计算模型中，实数的所有基本算术运算都只需单位时间完成。从组合的角度讲，强多项式算法更自然，因为只有它们的算术操作复杂度依赖于输入的数字大小，而其他操作的数量则与该大小无关。

第一个最大流问题的专用算法为 Ford 和 Fulkerson 设计的增广路径法。[14] 这种方法的时间一般不会是多项式时间，但却可以通过某种方式实现多项式时间。方法之一是通过 Dinic 论述的缩放。[11]

Edmonds 和 Karp[12] 引入了最短增广路径算法，把 Ford-Fulkerson 算法变成了强多项式。为了定义路径长度，Edmonds-Karp 算法使用了单位长度函数，把每条弧的长度设为一。Edmonds 和 Karp 注意到，也可以使用其他的长度函数，但似乎不会产生更佳的时间复杂度上界。（整个算法）分析的关键在于他们观察到，最短增广路径的长度不会减小，且最终必将增加。

阻塞流算法通过利用找出阻塞流的手段沿着最短路径的最大集进行增广。这种增广方式增加了最短增广路径的长度，进而加快了最短增广路径法的速度。在 Dinic 的算法中 [10] 隐含了阻塞流的思想。Karzanov 则明确提出了阻塞流，[23] 他还引入了名为“预流（preflow）”的流松弛。“预流（preflow）”允许算法改变单条弧中的流量，而不需要改变整条增广路径中的流量。弧的流量通过压入操作更新。预流支持用更快的算法找出阻塞流。

关于最大流问题，存在一个有趣又特别的例子，其中所有的弧均拥有单位容量。在 Karzanov[22] 以及 Even 和 Tarjan[13] 的论文中，他们独立的发现，基于以下两点原因，上述场景中阻塞流算法的时间复杂度比一般情况要好：阻塞流计算的数量减少；且计算更快，其运行时间与图的大小成正比。

阻塞流算法的操作可以分为两部分：处理距离的部分和处理流的部分。从理论方面来说，后者占据支配地位，其驱动了研究人员开发各种数据结构，获取更有效的方法改变路径中的流量值，而不是一次仅改变一条弧。Galil 和 Naamad[15] 设计了第一种该类型数据结构。几年后，Sleator 和 Tarjan[29,30] 引入了动态树数据结构，允许以 $O(\log k)$ 的时间修改由 k 条弧组成的路径中的流量值。这让寻找阻塞流的理论时间复杂度又前进了一步，把它几乎变成了线性时间。

Goldberg 和 Tarjan[18] 设计了压入 - 重标记法作为阻塞流法的替代方法。[a] 这种方法维护了一个预流，通过压入操作更新预流。它还引入了重标记操作来对顶点距离进行细粒度的更新。压入和重标记操作在本地执行；也就是说，它们分别应用于单一的弧和顶点。这些细粒度的操作提供了更高的灵活度，可用于设计更快的算法。最快的通用型最大流代码便基于压入 - 重标记法。[7,16]

对于任意实数值的容量，阻塞流问题能够在 $O(m \log(n^2/m))$ time,[19] 的时间内解决，进而给出了时间复杂度是 $O(nm \log(n^2/m))$ 的最大流算法。注意，把流分解成路径后，大小可能是 $\Omega(nm)$ 量级的。这样，$O(nm)$ 便成了一个自然的目标复杂性系下界。2013 年，Orlin[27] 设计了一个达到该下界的算法。

流分解的大小并非计算最大流的下界。一个流可以用 $O(m)$ 的空间存储，且可使用动态树以对数时间增大路径中的流。不仅如此，在无平行弧的图中，单元容量问题可以在 $O(\min(n^{2/3}, \sqrt{m})m)$ 时间内求解[13,22]，这比 $O(nm)$ 要好得多。在长达二十五年的时间内，单元容量场景和通用场景之间存在着巨大的差距。Goldberg 和 Rao[17] 缩小了这一差距，他们设计了时间复杂度为 $O(\min(n^{2/3}, \sqrt{m})m \log(n^2/m)) \log U)$ 的算法用于求解整数容量的问题。

为了实现这一复杂度上界，Goldberg 和 Rao 使用了非单元的长度函数。结合新的设计和分析技术后，它引发了可实现上述复杂度的二进制阻塞流算法。如同其名字一样，该算法基于阻塞流。而对于压入 - 重标记算法，尚未得到类似的结果。这一事实再次说明了阻塞流方法在理论上的重要性。

在本文中，我们假设读者已经熟悉了基本的图算法，包括广度和深度优先搜索算法。然后，我们按照下列顺序组织文章：先介绍基本定义，然后讨论算法。我们的表述通俗易懂，其中直观地描述了算法以及相应的时间边界，但省略了技术细节。有关技术细节的情况可参阅相关的参考文献。

最大流最小割定理指出，最大流的值与最小割的容量相等。

背景

最大流问题的输入为 (G, s, t, u)，其中 $G = (V, A)$ 是顶点集为 V 和弧集为 A 的有向图，$s \in V$ 为源点，$t \in V$ 汇点（但 $s \neq t$），且 $u: A \Rightarrow \mathbf{R}^+$ 为严格大于零的容量。有时我们假设容量为整数，并把最大容量记为 U。

流 f 是 A 的函数，其满足对所有弧的容量约束，以及对除 s 和 t 之外的所有顶点的守恒约束。对于 $a \in A$，容量约束为 $0 \leq f(a) \leq u(a)$（流量不得超过容量）。v 的守恒约束为 $\sum_{(u,v) \in A} f(u, v) = \sum_{(v,w) \in A} f(v, w)$（入流量等于出流量）。流量值为流入汇点的净流量：$|f| = \sum_{(v,t) \in A} f(v, t) - \sum_{(t,v) \in A} f(t, v)$。如果 $|f|$ 的值已经变得尽可能得大，则 f 为最大流量。一个割为顶点 $S \cup T = V$ 的二

a 压入 - 重标记法有时也成为预流 - 推进法，但这却容易让人误解，比如 Karzanov 的算法使用了预流和压入（推进）操作，但却没有使用重标记操作，因此它不属于压入 - 重标记算法。

分，其中 $s \in S$，$t \in T$。[b] 割的容量通过 $u(S,T)=\sum_{v\in S,w\in T,(v,w)\in A}u(S,T)$ 定义（从 S 到 T 的各弧的总容量）。最大流 / 最小割定理 [14] 指出，最大流的值等于最小割的容量。图 1 和图 2 分别描绘了输入网络及其中的最大流。

在不失去一般性的情况下，我们假设 G 是连通的。那么 $m \geq n-1$ 因此 $n+m=O(m)$。我们还假设，图中不包含平行弧，因为我们可以合并平行弧并把他们的容量相加。

剩余图和增广路径

流算法中的一个重要的概念是剩余图，它对弧上可能发生的流量变化进行编码，以便于算法设计。设我们有一弧 $a=(v,w)$，其中 $u(a)=9$ 和 $f(a)=4$。然后，我们在不违反容量约束的情况下 a 上的流量增加最多五个单位。接下来，我们把 a 上的流量减少最多四个单位。我们希望把弧 $a=(v.w)$ 上的流量减少解释为反向弧 $a^R=(w,v)$ 上的流量增加。

给定 G 中的流 f，剩余图 $G_f=(V,A_f)$ 的定义如下：A_f 包含弧 $a \in A$，满足 $f(a)<u(a)$；以及弧 $a^R: a \in A$，满足 $f(a)>0$。我们把它们分别称为前向剩余弧和反向剩余弧。我们把前者的剩余容量 u_f 定义为 $u(a)-f(a)$，后者的剩余容量定义为 $f(a^R)$。对于每一条弧 $a \in A$，G_f 包含前向弧、反向弧或两者。图 3 描绘了图 2 中的流的剩余图。注意，即使输入图相当简单，剩余图也可能包含平行弧，因为它可能同时包含一条弧及其反向弧。

通过 G_f 中的流 g，我们将 G 中的流 f' 定义如下：对于前向弧 $a \in G_f, f'(a)=f(a)+g(a)$；对于反向弧 $a \in G_f, f'(a^R)=f(a^R)-g(a)$。我们可以发现，$f'$ 是有效流的这一情况简单明了。

增广路径为 G_f 中从 s 到 t 的一条路径。给定一条增广路径 P，我们可按下列方式增广 f：设 δ 为 P 上各弧的最小剩余容量，g 为 P 上的流量值。G 上对应的流 f' 有 $|f'|=|f|+\delta>|f|$。增广路径可以在 $O(m)$ 的时间内找出（比如通过使用宽度优先或深度优先搜索）。

注意，在增广的过程中，P 中至少有一条弧的剩余容量 δ 在增广前大于零，在增广后等于零。我们把这种类型的弧称之为因增广而饱和的弧。饱和弧会被从 G_f 中删除。对于弧 a，如果增广前 $u_f(a)$ 为零，增广后 a^R 上的流量增加，则把该弧加入 G_f 中。

使用最大流 / 最小割定理后，你可以发现，当且仅当 G_f 不包含增广路径时，流 f 拥有最大值。这是增广路算法的基础：当 G_f 包含增广路径时，找出该路径并增大其上的流量。

若容量为整数，则增广路算法总能终止，因为每次增广至少让流量值加一。基于这一观察以及割 $(\{s\}, V-\{s\})$ 的容量为 $O(nU)$ 的事实给算法的运行时间设定了伪多项式上界 $O(nmU)$。该上界不是多项式的，因为 U 可能随问题的输入大小呈指数增大。如果容量是实值，则算法可能终止不了。在后面的讨论中，我们会发现，该算法的各种变体确实能以多项式时间运行。

缩放

当容量为整数时，缩放是让增广路算法达到多项式时间的一种手段。

回顾一下，U 指最大的弧容量；设 $k=\lceil \log_2 U \rceil+1$，指表示最大容量所需的比特数。对于 $i=0,\ldots,k$，定义 $u_i(a)=\lfloor u(a)/2^{k-i} \rfloor$。注意 $u_0 \equiv 0$；对于 $i>0$，定义 u_i 为 u 的 i 位最高有效位。零流量为 u_0 时的最大流量。

给定容量 u_i（$0 \leq i < k$）的最大流 f_i，算法计算容量 u_{i+1} 的最大流 f_{i+1} 的方法如下。注意，$u_{i+1}=2u_i+b_{i+1}$，其中 $b_{i+1}(a)$ 是 u（a）的第（$i+1$）位最高有效位。因此可以得出，$f=2f_i$ 是容量 u_{i+1} 的可行流。我们从 f 开始，然后应用增广路算法计算 f_{i+1}。

为了约束增广的数量，我们考虑了容量 u_i 的最小割（S,T）。因为 f_f 是最大流，对于从 S 到 T 的每一条弧 a，我们有 $u_i(a)=f_i(a)$，所以对于初始流 f，我们有 $u_{i+1}(a)-f(a) \leq 1$。因此，$|f|$ 位于最大值 m 的范围内，而且，从 f_i 计算出 f_{i+1} 时，我们最多需要进行 m 次增广来。所以，可以得出缩放算法的运行时间为 $O(m^2 \log U)$。

图 1 输入示例

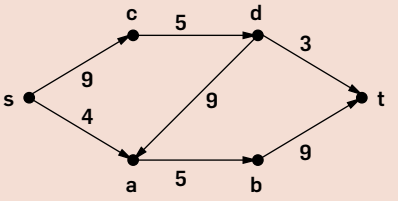


图 2 最大流（容量 / 流量）和最小割

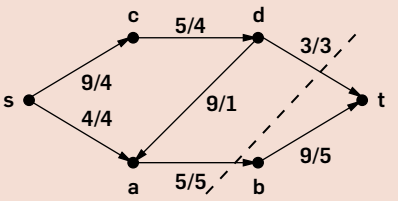
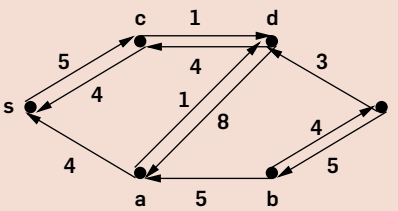


图 3 对应图 2 的剩余图和剩余容量



最短增广路径

定义 G_f 中每条边的长度为一，且假设我们总是选择最短增广路径。这很自然，因为广度优先搜索能以线性时间找出最短增广路径。

让我们来考虑一种最短路增广。设 $d(v)$ 为 G_f 中顶点 v 到 t 的距离，且设 $k=d(s)$。对于增广路径中的弧 (v,w)，我们有 $d(v)=d(w)+1$。因此，反向弧 (w,v) 并不在长度为 k 或小于 k 的 s-t 路径之内。增广会删除 s-t 路径中至少一条长度为 k 的弧，且不会向 s- t 路径添加任

b 从形式上来说，这定义了 s-t 割，而且本文中我们也只处理了 s-t 割；但是，在研究文献中，最小割也可以指在多个最小 s-t 割中所有 s,t 对的最小割值。

何长度为 k 或小于 k 的弧。这一观察引导人们发现了最短增广路算法的关键单调性质：对于每个顶点 v，s 到 v 以及 v 到 t 的剩余图距离非递减。

基于该单调性质，人们获得了强多项式时间边界。每次增广会让当前最短长度路径中的一条弧变成饱和。因此，最多经过 m 次增广后，s 到 t 的距离必定增大。初始时，距离至少为一。若 t 从 s 可达，则距离最多为 $n-1$。因此，增广的总数量为 $O(nm)$。用于找出增广路径的一次增广的时间为 $O(m)$，且该时间与路径长度成正比；换言之，更改流只需 $O(n)=O(m)$。这使得运行时间具有 $O(nm^2)$ 的上界。

阻塞流法

给定弧容量不为零的网络 G，如果 G 中每条 s 到 t 的路径中均包含饱和弧，则 G 中流 f 为阻塞流。注意，f 不必是最大流，因为 G_f 中可能存在 s 到 t 的路径，其中会包含 G 中弧的反向弧（见图 4）。但是，最大流总是阻塞流。在后面的讨论中，我们会看到，在无环图中，找出阻塞流的速度比最大流快。

图 4 阻塞流不是最大流的示例

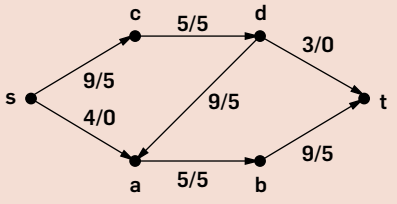
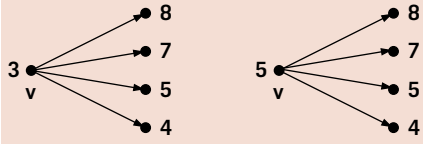


图 5 压入操作的示例：压入前（左）和压入后（右）；未标出零盈余量和非剩余弧。

图 6 重标记操作的示例：重标记前（左）和重标记后（右）。



阻塞流算法构建了辅助网络 $G'_f=(V,A'_f)$，其中 A'_f 包含了某条最短 s-t 路径中的所有剩余弧。注意，若 $(v,w)\in A'_f$，则 $d(v)=d(w)+1$，所以 G'_f 无环。通过广度优先搜索，G'_f 能以 $O(m)$ 的时间构建。假设我们计算出了 G'_f 中的阻塞流 g，则 $f+g$ 为 G 中的可行流。不仅如此，人们还会发现，在 G_{f+g} 中 s -t 的距离比 G_f 中的要大。由此推断，最多需进行 $n-1$ 次迭代便可计算出最大流，其中迭代的时间主要由阻塞流的计算确定。

Dinic[10] 介绍了在无环图中寻找阻塞流的算法，其先使用深度优先搜索找出 G'_f 中的增广路径，然后沿着路径进行增广，并从 G'_f 中删除饱和弧。分析的关键点基于如下观察：如果深度优先搜索从顶点回退，则在 G'_f 中从顶点到 t 不存在路径，可删除该顶点。人们可以利用这一观察说明算法的运行时间与 n 与发现的增广路径的总长度之和成正比；总长度的因素起决定作用。由于增广路径有 $O(n)$ 条弧，且每次增广会让一条弧变成饱和，所以阻塞流算法的运行时间为 $O(nm)$。该结论为 Dinic 的最大流算法设定了 $O(n^2m)$ 复杂性上界。

使用动态树

阻塞流算法的运行时间主要由在弧不饱和时对弧流量的变更决定。改进运行时间的边界时，一个自然的方法是使用某种特定的数据结构，允许在一次数据结构的操作中进行多项变更。通过能够记住增广路径的非饱和部分的数据结构，可以实现这一功能。为实现这一功能，研究人员设计了动态树数据结构[29,30]。

通俗地说，动态树阻塞流算法使用了可以记住增广路径非饱和部分的数据结构，支持随后对其进行重用。特别是在对增广路径进行搜索的过程中，保存的路径被链接在一起，这样工作便在增广路径的搜索之间得以分摊。流增广通过动态树操作执行，其运行时间为对应增广路径长度的对数。上述动态树的应用把阻塞流算法的运行时间从 $O(nm)$ 降低到了 $O(m\log n)$。限制树的最大规模并使用其他数据结构后，这一上界可以继续改进至 $O(m\log(n^2/m))$，[19] 由此我们便可得到复杂度为 $O(nm\log(n^2/m))$ 的最大流算法。

虽然动态树在最坏情况下仍能提供最佳的上界，但是他们却尚未在实际应用中使用，因为大多数实际的例子较为简单，而且动态树实现中的常数因子相对较大。

压入 - 重标记法

阻塞流算法使用了全局操作（例如构建辅助网络，沿路径增广）。压入 - 重标记法使用了本地操作。这些细粒度的操作不仅让方法更具灵活性，还能在实践中提高方法的速度。

按照 Karzanov[23] 的阐述，压入 - 重标记法使用了预流。预流与流类似，但是松弛了守恒约束：对于所有 $v\in V-\{s,t\}$，$\sum_{(u,v)\in A}f(u,v)\geq\sum_{(v,w)\in A}f(v,w)$（入流量大于等于出流量）。我们把盈余量定义为 $e_f(v)=\sum_{(u,v)\in A}f(u,v)-\sum_{(v,w)\in A}f(v,w)$。有盈余量的顶点可以把部分盈余量推向其剩余的邻居。凭直观而言，我们希望只把它推向较靠近汇点的邻居。Karzanov[23] 使用了辅助网络中的距离来确定应该把流推向何处。

压入 - 重标记法用有效的标记（即可在本地更新的距离的松弛）替代了距离。给定流 f 后，当 $d(t)=0$ 且对于每条 $(v,w)\in A_f$，我们有 $d(v)\leq d(w)+1$ 时，我们说函数 $d:V\to\mathcal{N}$ 是有效的标记。人们可以说明，使用有效的标记后，可以得到至 t 的距离的下界。特别是，若 $d(v)\geq n$，则 G_f 中无从 v 到 t 的路径，也就是说 v 位于某个最小割的源点侧。

压入 - 重标记法维护了一个预流 f，一个有效的距离标记 d，并分别通过压入和重标记操作对它们进行更新。后面我们会讲述这些操作；关于该算法的详细描述，参见 Goldberg 和 Tarjan[18] 的著作。

开始压入 - 重标记法的方式之一如下：对于从源点发出的所有弧，把它们流量值设为对应的容量值，让所有弧都变成饱和，而且如果 $v \neq s$，设 $d(s) = n$（如果无法从 s 出发到达 t）和 $d(v) = 0$。这样便在靠近源点的顶点上创建了初始的盈余量。直观来说，此方法会把盈余量推向汇点，而且如果盈余量无法被推向汇点，则用盈余量重标记顶点。当 $v \neq t$ 且 $e_f(v) > 0$ 时，我们说顶点 v 是活跃的。

如果 v 是活跃的且 $d(w) < d(v)$，或者据 d 判断 w 更靠近 t，则在弧 (v, w) 上执行压入操作。该操作确定了可被推进的最大流量值 $\delta = \min(e_f(v); u_f(v, w))$，并通过设置 $u_f(v, w) = u_f(v, w) - \delta, u_f(w, v) = u_f(w, v) + \delta, e_f(v) = e_f(v) - \delta$ 和 $e_f(w) = e_f(w) + \delta$，把该流量值沿 (v, w) 推进。如果 $\delta = u_f(v, w)$，我们说压入是饱和压入；否则我们说压入是不饱和压入。注意，非饱和压入丢弃了 v 的所有盈余量；图 5 说明了有关非饱和压入操作的例子。

重标记操作应用于活跃顶点 v，以便确保无压入操作用于弧 (v, w)，或确保对于所有 $(v, w) \in A_f, d(v) \leq d(w)$。该操作设 $d(v) = \min\{n, 1 + \min_{(v,w) \in A_f} d(w)\}$。（拥有盈余量的顶点总是会有一条流出的剩余弧）。注意，重标记操作总会增大 $d(v)$；图 6 描绘了重标记操作的一个例子。

压入 - 重标记法的时间复杂度如下：重标记的总时间为 $O(nm)$，饱和压入的时间也是 $O(nm)$。非饱和压入的时间为 $O(n^2m)$。因为这些操作决定了运行时间的边界，所以运行时间的复杂度也是 $O(n^2m)$。

注意，我们对压入 - 重标记法的描述是通用的：我们没有确定特殊的规则用于选择下一个待处理的活跃顶点。另外，采用某些特定的操作顺序后，可产生更佳的边界。特别是，对于采用最高标记的压入 - 重标记算法，它总是会选择拥有最高距离标记的活跃顶点向后处理，其非饱和压入的时间以及总时间的边界均为 $O(n^2\sqrt{m})$。[6] 使用动态树后，人们可以用比阻塞流更简单的方法获取时间为 $O(nm\log(n^2/m))$ 的边界 [18]。

关于最大流问题，存在一个有趣又特别的例子，其中所有的弧均拥有单位容量。

最高标记算法也是压入 - 重标记方法中实际价值最高的变体之一。不过，需要其他的求解规则来确保鲁棒的实际性能。压入 - 重标记法非常灵活，可使算法设计师能够轻松的添加求解规则。例如，人们可以把活跃顶点限制在 $d(v) < n$ 的顶点范围内，然后进行后处理来计算最终的流。人们还可以定期执行逆向广度优先搜索获取最大的 $d(v)$ 值。详细情况参见 Cherkassky 和 Goldberg 的著作 [7]，以及 Goldberg 其他的著作 [16]。

单位容量

现在，我们来考虑下一个最大流问题的特殊场景，其中所有的输入弧容量为一。因为合并平行弧会导致出现非单元容量，但我们又不能假设图中没有平行弧，所以我们考虑了两种情况——平行弧和非平行弧——在两种情况下，人们都能得到比 Dinic 算法更好的边界。

首先，我们注意到，在增广后，增广路径中的所有弧都已经饱和。因此，对于每个阻塞流而言，一条弧最多参与一次增广，而且阻塞流算法的运行时间为 $O(m)$。

不仅如此，人们还能证明，阻塞流的计算步骤数为 $O(\sqrt{m})$。为了证明这一边界，我们把最大流的计算分为两个阶段。在第一个阶段，s-t 距离小于 $\sqrt{m}$。因为通过阻塞流实现的增广增加了距离，所以第一个阶段最多包含 $\sqrt{m}$ 次增广。人们能发现，若 s-t 距离至少为 $\sqrt{m}$，则剩余流的值为 $O(\sqrt{m})$。因为增广减少了该值，所以第二阶段的增广次数为 $O(\sqrt{m})$。

上述分析意味着，对于单元容量问题，存在值为 $O(m^{3/2})$ 的复杂性上界。如果 G 没有包含平行弧，人们也能得到 $O(n^{2/3}m)$ 的上界。对于稠密图来说，情况会更好。

二进制阻塞流算法

一般情况下，阻塞流和压入 - 重标记算法的时间复杂度均为 $\Omega(nm)$。

相比之下，在单元容量的情况下，Dinic 算法的运行时间为 $O(\min(n^{2/3}, \sqrt{m})m)$。这里，我们会讨论隐藏在 Goldberg 和 Rao[17] 开发的二进制阻塞流算法背后的直觉。在整数容量的情况下，该算法缩小了差距。

二进制阻塞流算法没有给每条剩余弧设定单元长度，而是使用零-一长度函数，把拥有大剩余容量的弧的长度置为零，把拥有小剩余容量的弧的长度置为单元长度。这样，拥有单元长度的弧为具有小剩余容量的弧，这样算法就能接近于单元容量的时间边界。

该算法保持了一个流 f，以及最大流的值与当前流的值 $|f|$ 之间的差值上界 F。该算法的处理过程分为几个阶段；在每个阶段中，F 按除以二的方式减小。在一个阶段中，F 的值保持恒定，直到该阶段接近结束时才会减少。由于阈值参数 Δ 为 F 的函数，所以它在阶段中也会保持恒定，可确定剩余弧是大还是小；大弧的剩余容量至少为 3Δ，而其他的弧则为小弧。

与单元长度函数的情况类似，我们定义辅助网络 G'_f 为基于最短 s-t 路径中的弧而推导得出的图。该算法会反复计算 G'_f 中的阻塞流，并在每次计算前更新 G'_f，直到 F 减少到其二分之一时止。当人们把流量增加 $F/2$，或 s-t 距离变得足够大时，F 的值会减少。与单元容量的情况类似，较大的 s-t 距离暗示剩余流的值存在边界。

采用上述方式使用二进制长度函数后，会引起两个必须解决的问题：首先，G'_f 不必是无环的（它可以包含由长度为零的弧组成的环）；其次，弧的长度可以从一减至零，但它的副作用是，在阻塞流增广后，s-t 距离可能无法增加。

为了处理第一个问题，算法对 G'_f 中的强连通分量进行了收缩，然后在由此得出的无环图中查找阻塞流；在计算中，如果流量值达到 Δ，则停止阻塞流的计算。人们可以证明，因为每个强连通分量均由大容量弧推导得出，所以总是能找到一条路径引导值为 Δ 的流。在每次迭代结束时，我们会扩大 G'_f，并把我们找出的流扩充到 G_f 中。人们可以证明，G'_f 中的阻塞流可以扩展为 G_f 中的阻塞流。这让我们有了两种类型的迭代：找出阻塞流的迭代和找出值为 Δ 的流的迭代。在前一种情况下，s-t 的距离增加；在后一种情况下，s-t 的距离不会减少。

为了处理第二个问题，人们发现，对于长度可能会减少的弧 (v, w)（特殊的弧），它们拥有下列属性：(v, w) 的剩余容量至少为 3Δ，(w, v) 的剩余容量至少为 2Δ。该算法会收缩此类弧，确保在通过阻塞流进行增广后，即使这些弧的长度减少，s-t 的距离也会增加。

使用动态树数据结构后，二进制流算法的运行时间为 $O(\min(n^{2/3}, \sqrt{m})m\log(n^2/m)\log U)$。对于无平行弧的单元容量问题而言，该运行时间不会超过已知最佳上界的 $\log(n^2/m)\log U$ 倍。

完全理解最大流问题的道路还很长，而人们也在继续发现和改进算法。

结论

正如本文中论述的那样，Orlin[27] 在 2013 年设计的算法达到了最大流问题的强多项式复杂度上界 $O(nm)$；同时，当 $m = O(n)$ 时，它还达到了 $O(n^2/\log n)$。该结果相当复杂，结合了最大流算法、最小费用流和动态连通性算法中的诸多理念。特别是 Orlin 使用了二进制阻塞流算法作为子程序。最大流问题是否存在 $O(nm)$ 复杂度算法这一问题长期以来一直悬而未决，他的结果解决了这一问题。不过，二进制阻塞流的算法上界暗示，可能存在复杂度为 $O(nm/n^{\epsilon})$ 的强多项式算法。

完全理解最大流问题的道路还很长，而人们也在继续发现和改进算法。下面我们想提四个已经有新结果产生的有趣方向：第一个方向是推广压入-重标记方法，使得顶点盈余量（入流量减去出流量）可为任意值——正数、负数或零。给定剩余弧 (v, w)，使顶点 v 的盈余量超过 w 的盈余量，此时通过增大流量使弧饱和或者使 v 和 w 的盈余量相等，人们可以使弧达到平衡。流量平衡算法[31] 首先会使用某种初始流（比如值为零的流），

然后不加思考地设定 s 处的盈余量为正无穷大，而 t 处的盈余量为负无穷大。接下来，再重复弧平衡步骤，直到所有这些步骤只能移动极小的流量；最后，对流量执行四舍五入，获取准确的最大流。虽然该算法的运行时间（$O(n^2m \log U)$）无法与最佳的算法相提并论，但是该方法相当简单，而且在扩展该算法后，可以获得极其简单和实用的算法来处理最大流算法的参数多态版本。[2,31]

另一种方法是 Boykov 和 Kolmogorov[5] 设计的算法，它为计算机视觉应用中的最大流问题提供了快速实用的算法，其改进了基本的增广路径法，通过双向搜索来找出增广路径，并结合了一个巧妙的方法保存之前搜索的信息，从而提高了后续搜索的速度。Boykov-Kolmogorov 方法没有在最短路径上进行增广，也未被证明是多项式的。不过，可以对它进行改进，使得它能找出确切的最短路径，变成多项式的，同时也不牺牲它的实际性能。事实上，在很多情况下，人们都对它进行了改进。由此产生的算法 [20] 以递增的方式计算最短增广路径，并使用了从之前的搜索中获取的信息。对于平面图和无向图，在采取特殊的技术后，研究人员已经可以获取快速的最大流算法；有关最近的结果请参见 Borradaile 和 Klein[3]，Borradail 等人 [4] 以及 Karger 和 Levine 的著作。[21]

在最近发表的一系列论文中（包括 Christiano 等人 [8]，Kelner 等人 [24]，Lee 等人 [25] 以及 Sherman[28] 的论文）研究了如何在无向图中找出近似最大流的问题（值为最大值的 $1+\epsilon$ 倍），最终得到了近似线性时间的算法。这些论文使用了线性代数技术和电流的理念。基于上述工作，Madry[26] 在 2013 年取得了突破性的进展，他设计了处理有向图中单元容量流的精确算法，运行时间达到了 $\tilde{O}(m^{10/7})$。这让问题的经典上界 $O(\min(n^{2/3}, m^{1/2})m)$ 又前进了一步，同时还暗示，有向图中精确容量最大流问题可能存在更好的上界。至于这些理念是否可以用于找出有向图中整数容量的精确最大流，却仍然是一个有趣的、尚待解决的问题。总而言之，在过去的半个多世纪里，最大流算法取得了诸多进展，以后也会继续进步。

鸣谢

Robert E. Tarjan 得到了美国国家科学基金会（项目号 CCF-0830676）和美国—以色列两国科学基金会（项目号 2006204）的资助。他的部分工作在访问斯坦福大学期间完成，得到了美国空军科学研究办公室多学科大学研究计划的资助。

参考资料

1. Ahuja, R.K., Magnanti, T.L., and Orlin, J.B.*Network Flows:Theory, Algorithms, and Applications*.Prentice-Hall, Inc., Upper Saddle River, NJ, 1993.
2. Babenko, M.A., Derryberry, J., Goldberg, A.V., Tarjan, R.E., and Zhou, Y. Experimental evaluation of parametric max-flow algorithms.In *Proceedings of the Sixth Workshop on Experimental Algorithms, Lecture Notes in Computer Science*.Springer, Heidelberg, Germany, 2007, 256–269.
3. Borradaile, G. and Klein, P.N.An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph.*Journal of the ACM 56,* 2 (2009), 1–34.
4. Borradaile, G., Klein, P.N., Mozes, S., Nussbaum, Y., and Wulff-Nilsen, C. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time.In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*.IEEE Press, New York, 2011,170–179.
5. Boykov, Y. and Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision.*IEEE Transactions on Pattern Analysis and Machine Intelligence 26,* 9 (2004), 1124–1137.
6. Cheriyan, J. and Maheshwari, S.N.Analysis of preflow push algorithms for maximum network flow.*SIAM Journal on Computing 18,* 6 (1989), 1057–1086.
7. Cherkassky, B.V. and Goldberg, A.V. On implementing push-relabel method for the maximum flow problem. *Algorithmica 19,* 4 (1997), 390–410.
8. Christiano, P., Kelner, J.A., Madry, A., Spielman, D.A., and Teng, S-H.Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs.In *Proceedings of the Annual ACM Symposium on Theory of Computing*.ACM Press, New York, 2011, 273–282.
9. Dantzig, G.B.Application of the simplex method to a transportation problem.In *Activity Analysis and Production and Allocation,* T.C. Koopmans, Ed. John Wiley & Sons, Inc., New York, 1951, 359–373.
10. Dinic, E.A.Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematical Docladi 11* (1970), 1277–1280.
11. Dinic, E.A.Metod porazryadnogo sokrashcheniya nevyazok i transportnye zadachi [Excess scaling and transportation problems].In *Issledovaniya po Diskretnoi*.Matematike Nauka, Moscow, Russia, 1973.
12. Edmonds, J. and Karp, R.M.Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM 19,* 2 (1972), 248–264.
13. Even, S. and Tarjan, R.E.Network flow and testing graph connectivity.*SIAM Journal on Computing 4,* 4 (1975), 507–518.
14. Ford, Jr., L.R. and Fulkerson, D.R.Maximal flow through a network.*Canadian Journal of Mathematics 8* (1956), 399–404.
15. Galil, Z. and Naamad, A. An $O(EV \log^2 V)$ algorithm for the maximal flow problem.*Journal of Computer and System Sciences 21,* 2 (1980), 203–217.
16. Goldberg, A.V. Two-level push-relabel algorithm for the maximum flow problem.In *Proceedings of the Fifth Conference on Algorithmic Aspects in Information Management, Volume 5564 of Lecture Notes in Computer Science*.Springer, Heidelberg, Germany, 2009, 212–225.
17. Goldberg, A.V. and Rao, S. Beyond the flow decomposition barrier.*Journal of the ACM 45,* 5 (1998), 753–782.
18. Goldberg, A.V. and Tarjan, R.E.A new approach to the maximum flow problem.*Journal of the ACM 35,* 4 (1988), 921–940.
19. Goldberg, A.V. and Tarjan, R.E.Finding minimum-cost circulations by successive approximation.*Mathematics of Operations Research 15,* 3 (1990), 430–466.
20. Goldberg, A.V., Hed, S., Kaplan, H., Tarjan, R.E., and Werneck, R.F.Maximum flows by incremental breadth-first search.In *Proceedings of the 19th European Symposium on Algorithms*.Springer-Verlag, Heidelberg, Germany, 2011, 457–468.
21. Karger, D.R. and Levine, M. Finding maximum flows in undirected graphs seems easier than bipartite matching.In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*.ACM Press, New York, 1997.
22. Karzanov, A.V. Tochnaya otzenka algoritma nakhojdeniya maksimalnogo potoka, primenennogo k aadache 'o predstavitelyakh' [The exact time bound for a maximum flow algorithm applied to the set representatives problem].In *Problems in Cibernetics 5* (1973), 66–70.
23. Karzanov, A.V. Determining the maximal flow in a network by the method of preflows.*Soviet Mathematical Dokladi 15* (1974), 434–437.
24. Kelner, A., Lee, Y.T., Orecchia, L., and Sidford, A. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations.In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*.SIAM, Philadelphia, 2014, 217–226.
25. Lee, T., Rao, S., and Srivastava, N. A new approach to computing maximum flows using electrical flows.In *Proceedings of the Annual ACM Symposium on Theory of Computing*.ACM Press, New York, 2013, 755–764.
26. Madry, A. Navigating central path with electrical flows:From flows to matchings, and back.In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*.IEEE Press, New York, 2013, 253–262.
27. Orlin, J.B.Max Flows in $O(nm)$ time, or better.In *Proceedings of the Annual ACM Symposium on Theory of Computing*.ACM Press, New York, 765–774.
28. Sherman, J. Nearly maximum flows in nearly linear time.In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*.IEEE Press, New York, 2013, 263–269.
29. Sleator, D.D. and Tarjan, R.E.A data structure for dynamic trees.*Journal of Computer and System Sciences 26,* 3 (1983), 362–391.
30. Sleator, D.D. and Tarjan, R.E.Self-adjusting binary search trees.*Journal of the ACM 32,* 3 (1985), 652–686.
31. Tarjan, R.E., Ward, J., Zhang, B., Zhou, Y., and Mao, J. Balancing applied to maximum network flow problems.In *Proceedings of the 14th European Symposium on Algorithms*.Springer-Verlag, Berlin, 2006, 612–623.

Andrew V. Goldberg（goldberg@microsoft.com）是加利福尼亚州山景城微软研究院硅谷实验室的首席研究员。

Robert E. Tarjan（ret@cs.princeton.edu）是新泽西州普林斯顿镇普林斯顿大学计算机科学的 James S. McDonnell 杰出教授，兼任加利福尼亚州山景城微软研究院硅谷实验室的客座研究员。

译文责任编辑：孙晓明

技术视角
实现数据复制的一致性

Philip A. Bernstein

如需阅读对应的论文，请访问 /10.1145/2632792。

云计算系统中的**数据**应该具有高可用性。换言之，无论你何时接入系统，你在系统中保存的数据都应随时可用。实现该功能的标准机制——*数据复制*——需要维护每个用户数据的多个副本。

复制数据本身并不能解决问题，因为可用的数据副本可能是过期的。具体的解释如下，假设系统维护了文件 x 的三个副本：x_1，x_2，和 x_3。假设 x_1 和 x_2 当前可用，而 x_3 不可用。如果程序更新 x，则系统会把新值写入 x_1 和 x_2，但由于 x_3 不可用，所以无法更新 x3。这不会引发问题，因为系统仍然有两个正确的副本。接下来，假设 x_1 和 x_2 发生故障，然后 x_3 得以恢复。现在，问题出现了。如果用户读取 x，系统能提供的最佳结果是返回 x_3 的值。但是该副本是过期的——它没有包含最近的更新值。

针对这一问题，早期的解决方案为多数一致。[2] 处理 x 的写入操作时，系统会分配一个单调递增的时间戳，然后在 x 的多数副本中写入 x 的值和时间戳。处理 x 的读取操作时，系统会读取 x 的多数副本，然后返回具有最大时间戳的值。由于任意两个多数副本的交集至少会包含一个副本，所以每次读操作能够返回之前在该数据上写入的值。

在上文的例子中，读取操作只读取了一个副本，并非三个副本中的多数。因此，可能无法返回 x 的最新状态。如果使用了多数一致，系统就不得不等待第二个副本变为可用，这样才能读够两个副本。接下来，它会返回在读到的两个副本中更接近当前时间的副本，因为该副本的值必定是最新的。

给定 R、W 和 N，下列论文的作者提出了一个公式用于计算读到的数据不属于最近 K 次写入值之一的概率。

Gifford[1] 把多数的概念扩展为加权多数，并将其称之为*仲裁*。每个副本都会被赋予权重。读取操作必须访问达到*读取仲裁*的副本——副本集合的权重至少达到 R。而写操作则必须更新达到*写入仲裁*的副本——副本集合的权重至少达到 W。设定 R+W 必须超过所有副本的总权重 N 后，我们可以确保每次读取均能读到在最近一次写入操作中写入的值。

例如，设系统保存了 x 的四个副本，即 x_1-x_4。我们可以把 x_1 和 x_2 的权重设为 1；若 x3 和 x4 位于更可靠的服务器中，则把 x_3 和 x_4 的权重设为 2。这样得到 N=6。如果读取比写入更频繁，那么我们可以设 R=3，W=4，这样读取执行的任务会比写入少。因为 R+W>N，所以每次读取均能读到之前在该数据上写入的值。

按理说，每次写入操作都必须更新数据的所有可用副本。但是，读者也必须读多个副本，这颇为麻烦，因为它增加了开销和延迟。如果写入操作经常执行很快，这会特别棘手，因为在这种情况下，即使每次读取操作没有读取仲裁，也极有可能读到最新的副本。如果读到过期数据的概率足够低，那么让读者读取少于仲裁数量的副本或许是可取的。实际上，某些云应用也正是这么做的。

直到最近，这种通过猜测实现的折中方法才有所改观。原来对于读到的数据的过期性或承受某些过期风险后可期望获得的延迟减少缺乏界定。后面的论文阐述了一项突破性进展，它抛开了猜测，用名为概率有界过期性的原理分析取而代之。给定 R、W 和 N，论文的作者提出了一个公式用于计算读到的数据不属于最近 K 次写入值之一的概率。为了计算读到的数据不是在最近 Δ 时间单位内写入的值的概率，他们使用了基于时间参数的蒙特卡罗模拟，且这些参数可以在运行的系统中通过测量得出。不仅如此，他们基于开源记录管理器实现了该机制，让这种分析变得切实可行，使得用户可以依据自己的判断对过期性和延迟进行权衡。而原先的猜测工作现在也变成了人们可以精心策划的目标。

参考资料
1. Gifford, D.K.Weighted voting for replicated data.*SOSP* (1979), 150–162.
2. Thomas, R.H.A majority consensus approach to concurrency control for multiple-copy databases.*ACM Trans.Database Syst.4*, 2 (1979), 180–209.

Philip A. Bernstein（http://research.microsoft.com/~philbe）是华盛顿州雷德蒙市微软研究院的杰出科学家。

译文责任编辑：陈海波

DOI:10.1145/2632792

利用 PBS 量化最终一致性

Peter Bailis，Shivaram Venkataraman，Michael J. Franklin，Joseph M. Hellerstein，Ion Stoica

摘要

数据复制造成人们需要在操作延迟和一致性之间做出根本性的权衡。在可用的一致性模型范围中较弱的一端是最终一致性，它对返回数据的过期性没有施加限制。不过，说来有趣，对于从业者而言，最终一致性在延迟和可用性方面的收益往往“足够好”。在本文中，我们解释了这一现象，并论证了下列观点：虽然实现最终一致性的系统只能提供较弱的保证，但是他们通常能够返回一致的数据。不仅如此，与实现强一致性的同类系统相比，它们的延迟更低。为了量化实现最终一致性的数据存储的行为，我们引入了概率有界过期性（PBS）。这种模型从版本和墙上时钟时间两方面为数据过期性提供了预期边界。我们为基于版本的过期性推导出了一个闭式解。而且，对于一大类采用仲裁复制的 Dynamo 风格存储系统，我们构建了实时过期性模型。使用 PBS 后，在互联网生产级负载下，我们对部分的、非重叠的仲裁系统测量了在延迟和一致性之间做出的权衡。我们采用量化的方式阐述了最终一致性系统呈现下列特点的实现方式和原因：在提供相当大的延迟收益的同时，它们往往能在几十毫秒的时间内返回一致的数据。

1. 概述

现代的分布式数据存储需要变得可伸缩、高可用且访问速度快。基于至少以下的两点原因，这些系统通常会把数据复制到不同的机器上，并且会越来越多地把数据复制到不同的数据中心上：首先，在组件出现故障时提供可用性；其次，通过使用多个副本来响应请求，提供更好的性能。配置和维护复制数据对应用和数据存储设计产生了重要影响。[1] 对于一大类的应用而言，大规模集群的性能举足轻重。在实践中，增加延迟往往会导致收入大幅减少。[22] 例如，在亚马逊公司，100ms 的额外延迟会导致销售额下降 1%，[15] 而进行谷歌搜索时如果出现 500ms 的额外延迟，则会导致对应的流量下降 20%。[16] 然而，降低分布式数据存储的延迟需要付出一定代价：如每次操作访问更少的副本，则会对可实现的语义保证造成不利影响。

为了提供可预测的低延迟，现代系统通常会避开保证读操作“强”一致性的协议（例如，单一副本假象），而倾向于选择通常以*最终一致性*呈现的“较弱”的语义。[1, 5, 7, 10] 这种最终一致性是现代存储系统提供的最弱的属性之一：在无新的写入操作的情况下，读取操作最终会返回最近的（单次或多次）写入操作的结果；除了这点之外，它对数据过期性不提供任何保证。[26] 根据这一定义，只要在未来的*某个时间点*上，该存储返回了最后写入的数据 [4]，那么返回时间有数周之久的数据的存储就是最终一致的，返回任意数据的存储也是一致的（比如，返回值一直是 42）。对于终端用户而言，由于几乎没有有用的语义，使用最终一致性的决策往往饱受争议。[12, 23, 24] 在现在提供最终一致性的很多生产存储中 [10, 14]，用户几乎无法洞悉其存储的行为或其数据的一致性。在复制配置多变的情况下，情况更为明显。不过，最终一致性系统的广泛部署暗示，应用往往可以容忍偶发的过期性，而且在很多情况下，数据更倾向于“足够新”。

在本文中，我们通过量化最终一致性在最终性和（非）一致性方面所达到的程度并解释原因，从而弥合了理论保证与当前实践之间的鸿沟。事实上，在最坏的情况下，最终一致性会导致数据过期性的程度没有任何边界。然而，接下来我们会说明，通常的情况往往有所不同。我们论文的核心为下列观测结果，给定特定的工作负载和部署环境后，可以为最终一致性构建模型，用于提供一致性的概率*期望*。因此，对于程度不同的确定性而言，依照其可能偏离强一致行为的程度，最终一致的存储系统可对此提供界定。我们陈述了用于此类边界的概率模型，称之为概率有界过期性，或 PBS。

为了利用 PBS 预测一致性，我们需要知道最终一致的系统返回过期数据的时间和原因，并了解如何量化其返回数据的过期性。在本文中，我们提出了一些算法和模型用于测量文献中常见的两种过期

本论文旧版的标题为《用于实用的部分仲裁系统的概率有界过期性》，曾在 *VLDB 2012* 中发表。[5] 本文受邀扩展版本的标题为《使用 PBS 量化最终一致性》，将会在 2014 年 *VLDB Journal* 的《VLDB 2012 最佳论文》中发表。[7] 本文的部分内容还发表在 *SIGMOD 2013* 的演示中，题为《PBS 的运用：使用一致性度量提升数据管理》。[6]

性度量指标：墙上时钟时间[21]和版本[27]。PBS 描述了这两种度量指标，提供了在写操作返回（（Δ, p）-语义 或是 “什么程度的最终才算最终一致性？）Δ 秒后，读到该写入值的概率，读到数据项（（K, p）-语义，或是 “什么程度的一致才算最终一致性？”）的最近 K 个版本之一的概率，以及经历两者结合（（K, Δ, p）- 语义）后的概率。 PBS 并未提出新的机制来加强确定性的过期性边界；[27]与此相反，我们的目标是提供一个新的视角来分析、改进和预测广泛部署的*现有*系统的行为。

在本文中，我们把 PBS 应用于仲裁复制的数据存储中，比如 Dynamo[10]以及基于 Dynamo 产生的几个开源系统。通过确保读取操作和写入操作中的副本集存在重合，仲裁系统确保了副本读写操作的强一致性。不过，利用*部分*（或非严格的）仲裁后，由于需要做出响应的副本得以减少，延迟得以降低。利用部分仲裁后，写入和读取的副本集不需要重合：给定 N 个副本，设读取仲裁和写入仲裁大小为 R 和 W，部分仲裁意味着 $R+W \leq N$。对于部分仲裁，我们推导出了用于 PBS（K, p）- 正规语义的闭式解，并使用了蒙特卡罗方法探索延迟与（Δ, p）- 正规语义之间的权衡。

最后，对于在生产中部署的 Dynamo 风格的数据存储系统，我们使用 PBS 研究了其在正常情况下（即无故障场景中）观测到的过期性。我们说明了写入延迟的高方差会如何导致更宽的不一致窗口。例如，在某个生产环境中，从旋转硬盘转换到固态硬盘后，由于写入延迟的均值和方差减少，预期一致性有了大幅提升（例如，要达到 99.9% 的一致性读取概率，等待时间为 1.85 ms，而不是 45.5 ms）。我们还对部分仲裁提供在延迟与一致性上的权衡做了定量观测。例如，在另一个生产环境中，PBS 的计算说明，对于 202ms 的不一致窗口而言，在第 99.9 个百分位处（230 至 43.3ms）出现了 81.1% 的复合读取和写入延迟提升（一致性读取的概率为 99.9%）。该分析帮助业界论证了性能收益，可促使运营商选择最终一致性；它还推动了其他终端用户应用的发展，应用的范围涵盖了从一致性监测到基于一致性的服务水平协议（SLA）和查询规划等众多领域。

2. 概率有界过期性

在通常定义中，最终一致的数据存储系统并未对返回数据的新近程度做出任何保证：“如果对象没有得到任何新的更新，那么最终所有的访问均会返回最后更新的值。”[26] *活跃度*是一个有用的属性，它保证某个好的东西最终会发生，但是它并不提供任何安全属性：在中间阶段，数据存储可能返回任何数据。“[4]很多真实世界中的最终一致性存储并未做出超过上述定义的任何保证，但是它们仍然得到广泛部署。在过去的十年中，它们的流行程度与日俱增（见第 3.2 节）。接下来我们会看到，大多数最终一致性存储系统使用的协议确实没有*强加*额外的保证，但在运行中它们可能*提供*了这些保证。

为了量化没有保证但又往往提供的语义，我们开发了用于推断一致性的概率框架，将其称为概率有界过期性，或 PBS。数据存储可选择提供的一致性模型范围广泛，从线性一致性，到因果一致性，再到最终一致性；如果给定的一致性模型没有得到保证，终端用户观测到该模型的可能性有多大呢？本文中我们研究了以过期性的形式呈现的两种经典（非）一致性的变体：版本和时间。我们开发了不同的 PBS 度量标准，用于对下列情况提供量化期望：存储系统返回最近 K 次写入（若 K=1，则为最近的）的版本之一；以及存储系统返回 Δ 秒之前的最新版本。最后，虽然 PBS 并不提供保证，但它仍然有助于推断和反思给定系统的行为，这点与用于衡量性能的现代 SLA 类似（见第 6 节）。

首先，我们需要为 “强一致性” 语义定义一个基线。我们研究的 Dynamo 风格存储系统允许我们在 “强” *正规*语义和最终一致性之间进行选择；然后，我们观测在什么时候所选的最终一致性与对应的 “强” 语义的行为类似。在分布式系统的文献中：[2]

定义 1：若满足下列条件，则说明*给定数据项*的读取操作符合正规语义：*如果某数据项的读取操作与写入操作没有重合（实际时间），则返回最后完成的写入结果；或者，如果某数据项的读取操作与至少一次写入操作重合（实际时间），则返回最后完成的写入结果，或是重合的某次写入操作的最终结果。*

据此，正规语义提供了一种假象，让人们觉得各复制数据项只有一个副本。但是，在出现并发的读写操作时，写入操作的值可能变为可见但随后 “消失”。尽管这种特别的、重合的情况有点让人难以摸透（参考线性一致性的定义[13]），但这种数据复制配置仍被广泛部署。

在我们把 PBS 应用于正规语义前，我们先概括下用于解释多版本过期性的语义：[2]

定义 2：若满足下列条件，则说明*给定数据项的读取操作符合 K* – 正规语义：*如果某数据项的读取操作与写入操作没有重合（实际时间），则返回的结果为最近 K 次已完成的写入值之一；或者，如果某数据项的读取操作与至少一次写入操作重合（实际时间），则返回的结果为最近 K 次已完成的写入值之一，或是重合的某次写入操作的最终结果。*

K – 正规语义可用于推断给定读取操作的*过期程度*，也可用于强加额外的一致性属性，比如单调读，即读操作的结果不会出现 “回到过去”。[5, 25]

现在，我们来介绍 PBS 原理的首个应用：给定不保证 K- 正规语义的系统，我们可以采用概率方法来推断它的语义：

定义 3：*若系统中每次读取操作返回 K- 正规语义的概率为 p，则该系统提供了*（K, p）- 正规语义。

这种修改简单又直接：给定已有的语义保证，我们可以考虑其概率版本，其中读取有可能会也可能不会遵守属性的规定。当然，上面只是一个定义，我们仍然必须确定如何实际提供 PBS 预测，这也是下文的重点。

除了考虑基于版本的过期性外，我们还可以考虑实际时间方面的过期性。我们会看到，消息传播和处理延迟能够影响跨时间的一致性，所以我们把正规语义进行了扩展，纳入了时间因素：

定义 4：*若读操作*的返回结果为最多 Δ Δ *单位时间前的最新写入值，或是最多* Δ *单位时间前已经开始但尚未完成的写入结果*，则说明*读取*遵守 Δ- 正规语义。

与上文类似，我们可以把这个定义扩展到概率场景中：

定义 5：*若系统中每次读取操作遵守 Δ- 正规语义的概率为 p，则说明该系统提供了*（Δ, p）- 正规语义。

虽然在本文中我们未考虑上述定义，但是可以同时考虑时间和版本过期性，我们将其作为术语（K, Δ, p）- 语义。[5,7]

3. 仲裁系统的背景

有了 PBS 度量标准后，我们可以更进一步，把他们应用于实际系统中，发挥它们的最大作用。在本文的研究中，我们探讨了 PBS 度量标准在仲裁（基于复制的数据存储系统）中的应用情况；它们代表了一大类广泛部署的分布式数据存储系统。不过，基于我们的一些成果，我们相信该方法也可应用于其他风格的基于复制机制的系统。接下来，我们会阐述仲裁系统的背景，并重点关注当前的实践。

3.1. 仲裁的基础：理论

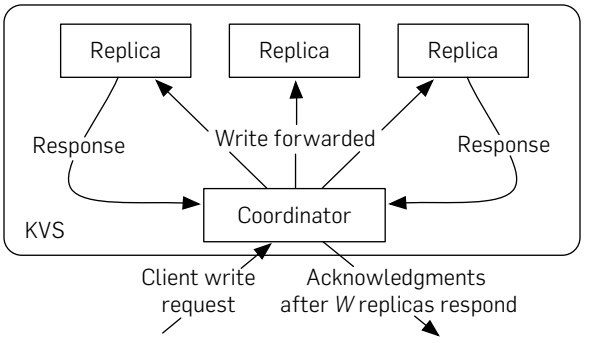
作为一种分布式数据的复制策略，仲裁系统有着很长的传统。[20] 在基于仲裁的复制机制中，当数据存储系统写入数据项时，会把数据项发给负责管理*副本*的服务器集合，我们将其称之为写入仲裁。为了提供读取操作，数据存储系统会从可能存在差异的副本集合中获取数据，我们将这种集合称之为读取仲裁。对读取操作而言，数据存储系统会使用全序版本信息比较副本返回的值集，并能够返回最近的值（或者如果需要，返回所有接收的值）。对于每个操作而言，数据存储系统从副本集合的某个集合中选择（读取或写入）仲裁，我们将其称之为*仲裁系统*，其中每个数据项拥有一个仲裁系统。仲裁系统有很多种类型，但是，对于大小为 N 的副本集而言，有一种简单的配置是使用固定大小的读取和写入仲裁，我们将其标记为 R 和 W。*严格的仲裁系统*有着仲裁系统中任何两个仲裁不重合（空交集）的性质，提供了正规语义。严格仲裁系统的一个简单范例是多数仲裁系统，其使用副本的数量作为仲裁 $\lceil \frac{N+1}{2} \rceil$。本文中将要研究*部分仲裁系统*，放宽了严格仲裁系统中约束：即在部分仲裁系统中，至少有两个仲裁间没有重合副本。[19]

3.2. 仲裁的基础：实践

在实践中，很多分布式数据管理系统利用仲裁作为复制机制。亚马逊公司的 Dynamo[10] 作为先驱引领了一类基于最终一致的数据存储系统，其中包括 Apache Cassandra[a]、Basho Riak[b] 和 Project Voldemort [c]。所有这些系统都使用了仲裁风格复制机制的相同变体，而且据我们所知，在广泛采用的数据存储中，各种存储使用的基于仲裁的复制协议不存在本质区别。

Dynamo 风格的仲裁系统为每个数据项设置了一个仲裁系统；通常情况下，会使用一致性哈希方案或集中式的成员协议对数据项在仲裁系统进行映射。系统集群中的每台服务器均保存有多个数据项。如图 1 所示，客户端向系统集群中的服务器发出读取和写入的请求，然后服务器会把该请求转发给操作涉及的数据项的*所有*副本。当协调服务器从预定数量的副本中收到响应后（通常基于每个操作进行配置），该服务器认为操作已经完成。因此，不会有消息丢失，所有的副本最终会收到所有的写入值。Dynamo 把数据项的复制因子设为 N，成功读取需要的副本响应数量为 R，成功写入需要的副本确认数量为 W。与其他严格的仲裁系统类似，在无故障操作期间，若 $R + W > N$，则 Dynamo 提供了正规语义。不过，与传统的仲裁系统有所不同，即使在操作返回后，Dynamo 的写入仲裁大小也会增加，其会通过*反熵*增大。[4,10] 协调器会把所有的请求发给所有的副本，但是只考虑最先的 R（W）个响应。基于命名方面的考虑（为了与“动态”仲裁成员协议加以区分），我们把这些系统称为*扩展型部分仲裁系统*。

图 1 客户端向 Dynamo 风格的仲裁进行写入时的控制流示意图（$N = 3$, $W = 2$）。协调器服务器处理客户端的写入操作，把它发给所有 N 个副本。当协调器收到 W 次确认后，写入调用返回。



[a] http://cassandra.apache.org/
[b] http://www.basho.com/riak/
[c] http://www.project-voldemort.com/

如同我们在本论文的扩展版本中的讨论一样，[5,7]，系统操作员往往会报告他们在 Dynamo 风格的存储中使用了部分仲裁配置，并提到在“通常情况”下达到了“最高性能”，特别是对于“低价值”的数据或“延迟极低且可用性高”的查询而言。

4.PBS 和部分仲裁

既然我们已经有了 PBS 度量标准，也了解了仲裁的行为，接下来我们可以构建多种模型分析部分仲裁中的一致性概率。在本文中，我们简要讨论了传统概率仲裁使用的基于版本的过期性，并开发了一个更为复杂的、基于时间的过期性的“白盒”模型，用于分析 Dyanamo 风格系统的基于时间的过期性。

4.1.PBS（K, p）- 正规语义

为了理解静态的、非扩展的仲裁行为，我们首先重温下概率仲裁系统 [19]。在部分仲裁系统中，它提供了仲裁交集的概率保证。举例而言，考虑一下 N 个副本的情况，其中读取和写入仲裁的大小分别为通过均匀随机抽取的 R 和 W。我们能够计算出读取仲裁未包含最后写入版本的概率。该概率的大小为，由写入仲裁中未写入值的副本组成的，大小为 R 的仲裁的数量除以可能的读取仲裁数量：

$$p_s = \frac{\binom{N-W}{R}}{\binom{N}{R}} \qquad 1.$$

如果 N 的值小，不一致性的概率会相当高。然而，通过缩放副本的数量和仲裁的大小，人们可以获得任意数值的高一致性概率。[19] 例如，设 $N = 3$，$R = W = 1$，$p_s = 0.\overline{6}$，但设 $N = 100$，$R = W = 30$，$p_s = 1.88 \times 10^{-6}$。[2]。这让人联想到了生日悖论：随着副本数量的增加，在任何两个仲裁之间不出现交集的概率减少。因此，这些系统的渐进性相当好——但是也只在渐进性的尺度上如此。

虽然概率仲裁让我们可以确定返回值为最近写入数据库的值的概率，但是它们却无法描述在未返回最近值时发生的情况。在本文中，我们确定了返回值属于限定数量的版本之一的概率（（K, p）- 正规语义）。在下面的阐释中，我们考虑了传统的、非扩展的写入仲裁（无反熵）。

与前面的示例类似，选定独立的、恒等分布（IID）的读取和写入仲裁后，返回值为最后 k 次写入版本之一等同于让 k 个独立的写入仲裁相交。假定单一仲裁不交的概率为 p，则其与最近 k 个独立仲裁之一不交的概率为 p^k。因此，在我们的示例仲裁系统中，与均匀随机抽取的选择不交的概率为等式 1 的 k 次方。

$$p = \left(\frac{\binom{N-W}{R}}{\binom{N}{R}} \right)^k \qquad 2.$$

若 $N = 3, R = W = 1$，这意味着返回值为 2 个版本之一的概率为 $0.\overline{5}$；返回值为 3 个版本之一的概率为 $0.\overline{703}$；5 个版本，>0.868；10 个版本，>0.98。若 $N = 3, R = 1, W = 2$（或者，等价的 $R = 2, W = 1$），这些概率增大：$k = 1 \rightarrow 0.\overline{6}$，$k = 2 \rightarrow 0.\overline{8}$，以及 $k = 5 \rightarrow >0.995$。

该闭式解对大小不随时间改变的仲裁成立。对于扩展的部分仲裁系统，该解为（K, p）- 正规语义的 p 的下界。在本论文的完整版本中，我们对本分析进行了更为深入的阐释，包括单调读的一致性分析，仲裁负载，以及时间和版本混合的闭式解。[5,7]

4.2.Dynamo 中的一致性

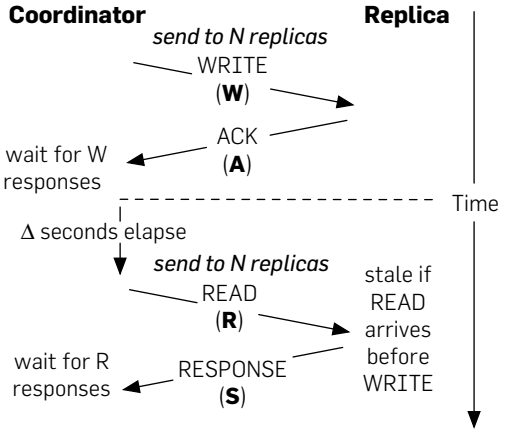
我们有一个简单的闭式模型用于（K, p）- 正规语义，但是基于时间（（Δ, p）- 正规）语义取决于给定系统采用的仲裁复制算法、工作负载和任意的反熵流程。在本节中，我们开发了多种技术在 Dynamo 风格的数据存储系统中分析 PBS（Δ, p）- 正规语义。

由于读写消息重排序及其造成的消息延迟等原因，Dynamo 风格的仲裁系统是不一致的。在本文中，我们采用了白盒方法分析不一致性，并直接检查一致性背后的协议。相应地，我们开发了 Dynamo 操作的消息延迟模型，用于捕捉写入请求（W）、写入确认（A）、读取请求（R）和读取响应（S）的消息延迟的影响。为方便起见，我们把它们统称为 *WARS*。在图 2 中，我们使用消息的时空图说明了 *WARS*。图中说明了在写入操作完成 Δ 秒后再进行读取操作的过程中，协调器和单一副本之间的消息传递情况。相应地，此处的 Δ 对应 PBS（Δ, p）- 正规语义中的 Δ。简言之，在最后（已完成）的写入请求到达前，如果所有最先的 R 均响应了到达它们的副本的读取请求，则读到的值是过期的。

对于写入操作而言，协调器会发出 N 条消息，每个副本一条消息。从协调器向副本发出的、包含写入操作的消息延迟程度为从分布中推导出的值 W。协调器等待从副本中发出的 W 次响应，然后才认为写入操作完成。确认写入操作的每条响应的延迟程度为从分布中推导出的值 A。

对于读取操作而言，协调器（可能与写入操作的协调器不同，且代表的客户端与发出写入操作的客户端也可能有所不同）发出 N 条消息，每个副本一条消息。从协调器向副本发出的、包含读取请求的消息延迟程度为从分布中推导出的值 R。协调器等待从副本中发出的 R 次响应，然后才返回它接收到的最新的值。

图 2 通过为消息延迟（执行写入操作 *t* 秒后再执行读取操作时，协调器和副本之间传递的消息）建模，*WARS* 模型描述了 Dynamo 中的过期性。在拥有 *N* 个副本的系统中，图中描绘的消息与 *N* 副本进行交流。



每个副本响应读取操作的延迟程度为从分布中推导出的值 S。

如果读取操作的协调器最先接受到的 R 次响应在副本接受最新的版本前到达副本（延迟程度为 W），则读取协调器会返回过期数据。如果 $R+W>N$，这不可能发生。然而，在部分仲裁中，上述情况的发生频率取决于延迟的分布情况。如果我们把写入操作的完成时间（若协调器已经收到 W 次确认）标为 w_t，对于从 R 中推导出的 r' 以及从 W 中推导出的 w'，若 $r'+w_t+\Delta<w'$，则单一副本的响应是过期的。在协调器等待所有所需的确认（A）和副本等待读取请求（R）时，写入操作有时间传播到更多的副本。在传回（S）读取操作的协调器的过程中，读取操作的响应又被延迟了，造成重排序的可能性增大。定性地来看，由于重排序的原因，呈长尾分布的写入操作（W）和相对较快的读取操作（R，S）会增大过期性的几率。

WARS 考虑了消息发送、延迟和接收的影响，但也提出了一个难以进行确切分析的问题，因为它带有几个非独立阶统计量。正如我们在第 5.1 节中讨论的那样，我们会使用蒙特卡罗方法探讨 *WARS*，因为理解和实现该方法相当简单。我们已经发现，给定真实世界系统的行为轨迹后，对 *WARS* 分布进行参数化相当容易（见第 5.3 节）。

5.PBS 实战

既然我们已经拥有了 Dynamo 风格存储的 PBS 模型，现在我们就把它们应用到真实世界系统中去。如同第 4.2 节中讨论的那样，给定系统中的 PBS (Δ,p)- 正规语义取决于读取操作和写入操作在副本间的传播。我们引入了 *WARS* 作为一种手段来推断 Dynamo 风格的仲裁系统中的不一致性，但在实践中观测到的定量度量指标（如过期性）取决于 *WARS* 的各种延迟分布情况。在本节中，我们依照人工创建的分布和真实世界的分布对 Dynamo 风格 (Δ,p)- 正规语义进行了分析，以便更好地理解“最终一致”意味着“一致”的频率有多高。更重要的是，还能让人更好地理解造成 Dynamo 风格的存储确实经常一致的原因。

部分仲裁的 PBS (K,p)- 正规分析可以较容易地以闭式的形式获取（见第 4.1 节）。它不依赖于写入操作的延迟或任何环境变量。实际上，在实践中如果没有扩展仲裁或反熵，我们观测到我们推导出的方程在实验中成立。

相比之下，(Δ,p)- 正规语义依赖于反熵，所以要复杂得多。在本节中，我们的重点放在 PBS (Δ,p)- 正规语义的实验期望的推导上。首先，我们基于 *WARS* 模型验证了我们的蒙特卡罗分析，其中使用了从伯克利的 Cassandra 集群中收集到的消息级轨迹。然后，我们探讨了人工创建的延迟分布。在剩余的分析中，我们探讨了源于下列两家互联网公司的分布：领英（LinkedIn）和 Yammer。

5.1. 蒙特卡罗模拟

我们在基于蒙特卡罗的模拟中实现了 *WARS* 分析。给定 Δ 的值后，计算 (Δ,p)- 正规语义简单又直接（见 Bailis 等人的伪码[7]）。为了模拟协调器与 N 个副本中每个副本之间的消息延迟，我们把从分布 D 中取出的第 i 个样本标记为 $\mathrm{D}[i]$，并从 W, A, R 和 S 中取出 N 个样本。然后计算写入请求完成的时间（w_t，或是协调器得到其第 W 个确认的时间；$\{\mathrm{W}[i]+\mathrm{A}[i],\ i\in[0,N)\}$ 的第 W 个最小值）。接下来，确定在最先的 R 个副本中，是否有任何副本包含最新的响应：检查 R 中最先的 R 个样本，依据 $\mathrm{R}[i]+\mathrm{S}[i]$ 排序，须符合 $w_t+\mathrm{R}[i]+\Delta\le\mathrm{W}[i]$ 的要求。多次重复这一过程后，我们可以获得对由轨迹确定的行为的近似。给定写入操作的到达时间分布，如果要把上述公式拓展到 (K,Δ,p)- 正规语义的分析中去，则需要考虑跨时间的多次写入操作。如同本论文的扩展版本中描述的那样，[5,7] 我们利用从真实世界的 Cassandra 集群中收集到的轨迹验证了上述分析。我们观察到，(Δ,p)- 正规语义的预测出现了 0.28% 的均方根误差，延迟预测出现了 0.48% 的平均标准均方根误差。

5.2. 写操作延迟分布的影响

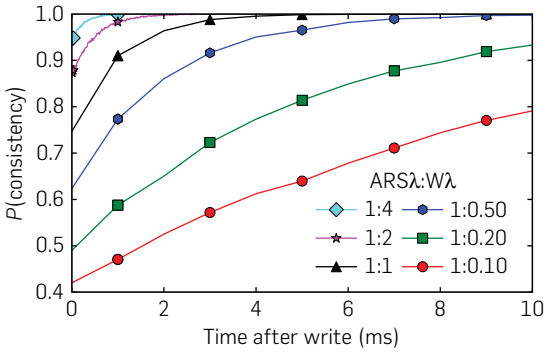
Dynamo 风格系统的 *WARS* 模型说明，延迟的高方差增大了过期性。在研究真实世界的工作负载前（第 5.3 节），我们用人工创建的分布来单独量化该行为：我们快速观察了很多呈指数分布的写入操作（改变参数 λ，因为它决定了分布的均值和尾），同时固定 $\mathrm{A}=\mathrm{R}=\mathrm{S}$。

我们的结果说明了这种关系，见图 3。当 W 的方差和均值为 0.0625 ms 和 0.25 ms 时（$\lambda=4$，$\mathrm{A}=\mathrm{R}=\mathrm{S}=1$ ms 时均值的四分之一），我们观察到，在紧接

着写操作后的一致性几率为 94%；而间隔 1ms 后的几率为 99.9%。然而，当 W 的方差和均值为 100 ms 和 10ms 时（$\lambda = 0.1$，A = R = S = 1 ms 时均值的 10 倍），我们观察到，在紧接写着操作后的一致性几率为 41%；只有在间隔 65ms 后的几率才为 99.9%。随着方差和均值增大，不一致性的概率也会增大。分布的均值固定但方差可变（均匀分布，正态分布）时，我们观察到，若 W 严格大于 A = R = S，则 W 的均值没有它的方差重要。

减少 W 的均值和方差会提高一致性读取的概率。这意味着，降低写入延迟的方差的技术可以导致更一致的读取操作，接下来我们将会看到这一点。与增加读取和写入仲裁的大小相反，操作员可以选择通过硬件配置或通过推迟读取操作降低（相对的）W 延迟。但是，对于读取操作占主要地位的工作负载而言，后者会损害性能并可能造成不良的排队效应。尽管如此，PBS 分析还是说明了一个事实，即除了简单地调整仲裁大小外，还有多种途径可以避免读取过期的值。

图 3(Δ, *p*)- 正规语义，其中 W 的延迟分布呈指数增长且 A = R = S。平均延迟为 1/λ。*N* = 3, *R* = *W* = 1。



5.3. 生产中延迟的分布

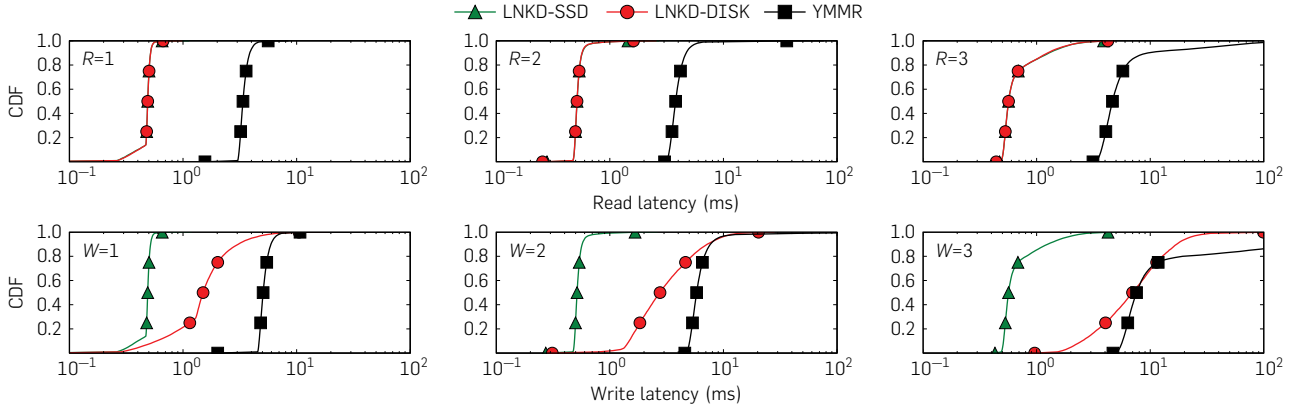
为了研究真实世界的行为，我们从两个互联网规模的公司获取了生产中的延迟统计数据。虽然消息级的 *WARS* 时间轨迹可以提供更为准确的预测，但是我们还选择了从务实的角度进行折中分析：正如我们在本文的扩展版本中描述的那样，我们让 *WARS* 分布变得适于分析所获取的各种统计数据（图 4）。[5,7]

领英（LinkedIn[d]）是一家专业性的在线社交网络。至 2013 年 7 月止，它拥有超过 2 亿 2 千 5 百万会员。为了提供高可用、低延时的数据存储，LinkedIn 的工程师搭建了 Voldemort。Alex Feinberg 是 Voldemort 的首席工程师。他慷慨地为我们提供了在处理 LinkedIn 中面向用户的服务时，单台服务器在高峰流量下的延迟分布数据，其中呈现了 60% 的读和 40% 的读 - 改 - 写流量。[5,7]。Feinberg 报告说，使用旋转硬盘时，Voldemort “在很大程度上受 IO 的限制，延迟大部分由我们使用的硬盘类型、数据与内存的比值以及请求分发决定”。使用固态硬盘（SSD）后，Voldemort 往往“受 CPU 和 / 或网络的限制”，而且“最大延迟通常由【垃圾回收】活动决定（垃圾回收极少发生，但偶尔也会发生），且时间在几百毫秒之内。”我们把 LinkedIn 使用旋转硬盘时的分布标为 LNKD-DISK，SSD 的轨迹标为 LNKD-SSD。

到 2013 年 7 月止，Yammer[e] 向 20 多万家公司提供了私有社交网络，它使用了 Basho 的 Riak 存储某些客户数据。基础设施架构师 Coda Hale 为我们提供了生产环境中的 Riak 部署的性能统计数据。[5,7]Hale 提到，“读取和写入拥有截然不同的预期延迟，对 Riak 而言情况更为明显。”Riak 会延迟写入，“直到 fsync 返回，所以读取通常 <1 ms，而写入极少 <1 ms。”此外，虽然我们没有明确地为其建模，Hale 注意到，值所占的

[d] http://www.linkedin.com/
[e] http://www.yammer.com/

图 4 生产中读取和写入操作的延迟，适合用于 *N* = 3 和变化的 *R* 和 *W*。对于读操作，LNKD-SSD 与 LNKD-DISK 等效。图中绘制的点标出了重要的百分位以便对比。较高的 *R* 和 *W* 值会导致延迟增加。



空间大小相当重要；他声称，“在对值进行 LZF 压缩后，获得了相当大的性能提升。”我们把 Yammer 的延迟分布标为 YMMR。

5.4. 生产中的过期性

生产延迟的分布确认了在使用最终一致性的存储中过期性往往会受到限制。我们测量了每个分布的（Δ, p）-正规语义（见图 5）。LNKD-SSD 和 LNKD-DISK 的数据说明了实践中写入延迟的重要性。紧接着写入完成后 LNKD-SSD 的一致性读取的概率为 97.4%；而间隔 5 ms 后的一致性读取的概率超过了 99.999%。在紧接着写入完成后 LNKD-SSD 的读取操作几乎能与写入操作平齐。然而，在写入操作后的若干毫秒内，读取操作在最后写入前到达的几率几乎为零。该分布的读取和写入操作延迟很低（中位数为 0.489 ms），而且由于该分布是短尾的（第 99.9 个百分位为 0.657 ms），所以跨所有副本的写入操作均能很快完成。相比之下，在 LNKD-DISK 数据中，写入操作的时间要长很多（中位数为 1.50 ms），而且存在较长的尾（第 99.9 个百分位为 10.47 ms）。LNKD-DISK 的（Δ, p）- 正规语义反应了这一差异：紧接着写入完成后 LNKD-DISK 的一致性读取的概率只有 43.9%；而间隔 10 ms 后的概率只达到了 92.5%。这说明，由于 SSD 降低了写入操作的方差，所以可能大大提升了一致性。与此类似，人们可能会期望，如果使用明确的内存管理来替代无法计划的垃圾回收，那么一致性可能会有所提升。

在另一个分布中，我们经历了类似的行为。紧接着写入操作完成后（$\Delta = 0$）YMMR 的一致性概率为 $p = 89.3\%$。不过，由于其写入分布存在高方差和长尾现象，在 $\Delta = 1364$ ms 时，YMMR 分布的一致性的概率仅达到 $p = 99.9\%$。这意味着，给定数据项的多个副本，同步把值写入磁盘能获得稳定性方面的收益，但却可能会对一致性产生不利影响。另一种提高一致性并避免高方差的方法是依赖多副本（内存或缓冲区高速缓存）复制实现稳定性，但只用异步的方式写入值。

5.5. 仲裁大小的调整

除了使用 N=3 ——即我们在实践中碰到的最常见的仲裁大小外——我们还考虑了，如果保持 R = W = 1 不变，改变副本的数量（N）会对（Δ，p）- 正规语义造成何种影响。图 6 描绘的结果说明了随着 N 的增加，紧接着写入完成后的一致性概率会随之降低。如果副本的数量为二，则紧接着写入完成后 LNKD-SSD 的一致性读取的概率为 57.5%；如果副本的数量为 10，则概率

图 5 用于生产操作延迟的（$^{\Delta}, p$）- 正规语义

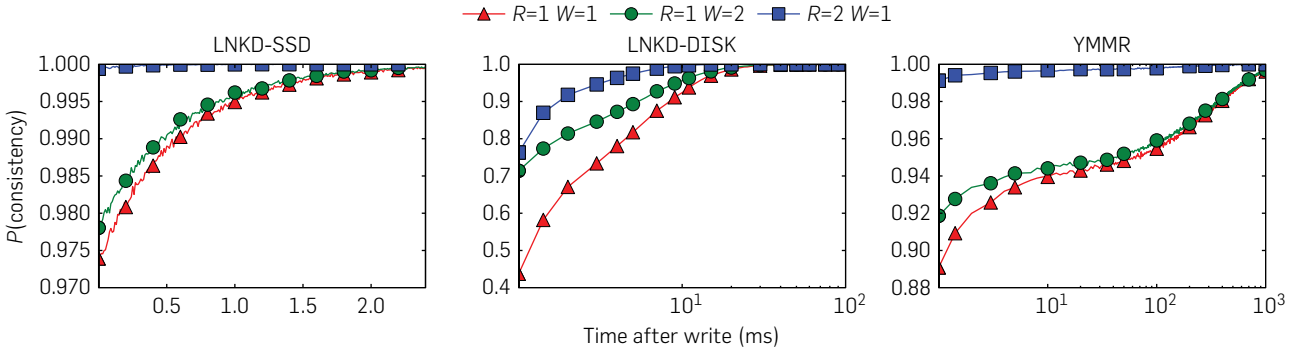
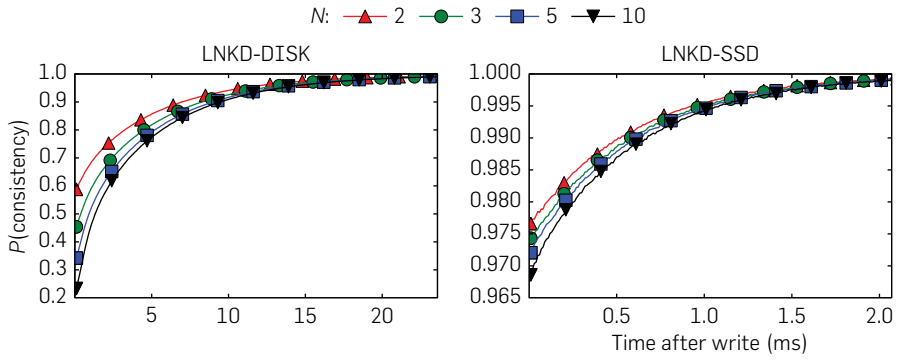


图 6 用于生产操作延迟的（$^{\Delta}, p$）- 正规语义，其中 $R = W = 1$，但副本的数量 N 不同即使复制的因子相当大，不一致性 p 也可能很快消解（$^{\Delta}$低）。



只有 21.1%。然而，在高概率（p）时，对于更大的副本规模，不一致的窗口（Δ）是关闭的。对于 LNKD-DISK，p = 99.9% 时，Δ的范围包括从 2 个副本时的 45.3 ms 至 10 个副本时的 53.7 ms。

这些结果说明，为获取可用性或更高的性能而维护大量的副本可能会对紧接着写入操作后的一致性造成相当大的影响。不过，（Δ, p）- 正规语义的概率（p）仍然会迅速上升（小 Δ）。

5.6. 延迟与过期性之比较

选择 R 和 W 的值时，需要权衡操作延迟和一致性。为了度量这种权衡，我们比较了第 99.9 个百分位的操作延迟和对应的 p = 99.9% 时的 Δ，其中仲裁配置为 N = 3，在实际中是一种典型的部署方式。

部分仲裁往往能呈现令人满意的延迟和一致性的权衡（见表 1）。对于 YMMR 而言，$R = W = 1$ 时，会获得低延迟的读取和写入（16.4 ms），但 Δ 相当高（1364 ms）。不过，设置 R = 2、W = 1 时，Δ 会降低到 202 ms，读取和写入的综合延迟比最快的严格仲裁（W = 1、R = 3）低 81.1%（186.7 ms）。如果允许 p = 99.9%、Δ = 13.6 ms，则 LNKD-DISK 读取和写入延迟会降低 16.5%（2.48 ms）。对于 LNKD-SSD 而言，在 10 M 次写入操作（“七个九”）中，取 R = 2、W = 1 时，我们没有观察到过期性。$R = W = 1$ 会把延迟降低 59.5%（1.94 ms），相应的 Δ= 1.85 ms。总之，降低 R 和 W 的值可以极大地改进操作延迟；而且，即使是在尾部，不一致的持续时间（Δ）也相对较小。

本文中，我们省略了完整的结果，但我们已经针对异构的副本行为进行了实验，也针对多条目的保证（如因果一致性和事务的原子性）进行了实验。[7]

6. 相关讨论和未来的工作

在本节中，我们讨论了 PBS 的设计决策，描述了我们如何将 PBS 与真实世界的存储和终端用户应用进行集成，并提出了未来的研究领域。

6.1. 预测和验证

在本文中，我们开发了多种技术用于一致性的预测。给定系统的输入数据集和当前的操作环境后，它们可以提供期望的系统行为。给定轨迹后，人们可以预测在任意长度的时间或版本数量后出现的过期性，而不需要实际执行额外的查询。不仅如此，预测还能让用户轻松地执行“假设”分析，获取在任意的复制配置、请求分发（Δ 和 K）和硬件配置条件下的结果（例如，从固态硬盘切换到旋转硬盘）。我们发现，预测的计算并不昂贵，它能以分散的方式基于每个副本的具体情况执行。虽然预测较为灵活，但它的好坏程度只能达到输入轨迹的好坏程度。如果使用代表性的输入数据，那么预测是准确的。然而，如果使用不具代表性的数据（或不良的模型），那么预测的准确度会受到影响。

相比之下，验证 [21] 会告诉用户，在确保了确定性时，他们的数据存储如何运行。如果用户使用预测器改变他们的副本设置，她可能想确保改变后的行为与预期一致。虽然此类验证并非特别适合“假设”分析，但它仍然是预测的一个重要补充。验证有效地提供了一种度量指标，这种指标源于集成（K, Δ, p）- 正规语义的密度函数，其中依据给定的读取请求的速率进行了加权（通过最后写入后经历的时间进行测量）。不仅如此，虽然验证在算法上相当复杂 [11]，但根据我们的经验，实现起来并不是遥不可及。

我们相信，随着各个系统开始把一致性当成持续性的量化指标，上述两种技术的作用将会越来越大。整体而言，一致性预测和验证技术组成了一个功能强大的工具包。

6.2. 白盒和黑盒方法

在本文中，我们采用了白盒方法分析不一致性，并利用复制协议的专业知识提供定量方面的洞察。这需要把以用户为中心的、声明式的一致性异常规范转变为后端的协议事件（例如，在 *WARS* 模型中，读取和写入响应

表 1（$^{\Delta}, p$ =99.9%）- 正规语义和第 99.9 个百分位处读取（L_r）和写入延迟（L_w）对应的 $^{\Delta}$ 值，其中 R 和 W 可变，N =3

	LNKD-SSD			LNKD-DISK			YMMR		
	L_r	L_w	t	L_r	L_w	t	L_r	L_w	t
R = 1, W = 1	**0.66**	**0.66**	**1.85**	0.66	10.99	45.5	5.58	10.83	1364.0
R = 1, W = 2	0.66	1.63	1.79	0.65	20.97	43.3	5.61	427.12	1352.0
R = 2, W = 1	**1.63**	**0.65**	**0**	**1.63**	**10.9**	**13.6**	**32.6**	**10.73**	**202.0**
R = 2, W = 2	**1.62**	**1.64**	**0**	1.64	20.96	0	33.18	428.11	0
R = 3, W = 1	4.14	0.65	0	**4.12**	**10.89**	**0**	**219.27**	**10.79**	**0**
R = 1, W = 3	0.65	4.09	0	0.65	112.65	0	5.63	1870.86	0

重要的延迟 - 过期性权衡用粗体标出。

的重排序）。另外，我们本来也可以尝试对系统内部进行反向工程，或提供一个不依赖于具体实现的预测器，但是这种黑盒分析将会变得极为复杂。我们相信，验证技术更适于黑盒技术，但必须对黑盒技术所带来的概率收益从其潜在的不准确性角度进行权衡。从我们把预测与现有数据存储进行集成的经验以及大规模采用开源数据存储的情况来看，我们相信白盒技术是可行的，即便他们需要对现有的存储进行修改。

6.3. 真实世界的存储集成

在几位开源软件开发者的帮助下，我们开发了在下列两个 NoSQL 存储系统中使用 PBS 功能的补丁：Cassandra 和 Voldemort。对于 Cassandra，我们采取了两种方法：一种是侵入式的、但更为准确的实现，另一种是外部的、但准确度稍低的预测模块。对于前一种方法，我们修改了 Cassandra 的消息层，加入了消息创建的时间戳，以便测量 W、A、R、S 的单独分布。在给定的服务器上启用跟踪后，消息层在独立的 PBS 预测模块中记录了每次操作的时间戳作为日志。时间戳储存在内存中的循环缓冲区内，用于处理各种所需的消息延迟。接下来，用户可以通过外部可访问的接口调用 PBS 预测器模块。利用该接口后，他们还能提供更高级的功能，比如动态副本配置和监测（见下）。这提供了相对准确的预测，其代价是必须调试消息层。不过，像 Cassandra 这样的数据存储系统已经扩展了用户可访问的监测数据（例如，每次查询的延迟跟踪）。最近，我们一直在探索在数据库之外进行预测——即后一种方法——我们实现并应用于对 Voldemort 的分析。我们为这两种方法都开发了开源的实现。

6.4.PBS 应用

如果没有一致性量化标准，某些功能就无法实现。而 PBS 让这些功能成为了现实。拥有 PBS 后，通过设定过期性和最低稳定性的约束，优化操作延迟，我们可以自动地配置复制参数。数据存储的操作员随后可以为应用提供服务水平协议，并用量化的方式向用户描述延迟 - 过期性之间的均衡关系。操作员还可以使用在线的延迟测量结果动态地配置副本。这种优化还支持人们分离用于稳定性的副本与用于实现低延迟和较高容量的副本。例如，操作员可以确定一个用于稳定性和可用性的最小复制因子，但也能自动增大 ***N***，减少 ***R*** 和 ***W*** 恒定后的尾延迟。在 ***SIGMOD 2013*** 的演示中，我们详细阐述了这些可能的方法。该演示的特点为对活跃的 Cassandra 集群和模拟 web 服务进行实时预测。[6]

6.5.*WARS* 的局限性和扩展

WARS 模型存在若干局限性，也有一些潜在的扩展。在本文中，我们大致勾勒了一些要点，更为详细的讨论请阅读本文的扩展版本。[5,7] ***WARS*** 只为单一的写入和读取操作建模，所以对多写入的场景而言，它的估计有点保守。不仅如此，在我们当前的处理中，***WARS*** 把每种分布当成 IID。对模型而言，这虽然不是根本性的，但却对我们从业界获取的延迟轨迹施加了限制。***WARS*** 也没有捕捉额外的、常用的反熵流程的效果（例如，读 - 修复，Merkle 树交换）。对于过期性而言，它的估计可能有点保守。它也没有处理发生故障时的系统行为；当存储不同时该系统行为也不同。而且，它假设客户端与协调器服务器联系，而不是自己发出请求（例如，Voldemort）。我们相信，这些局限性并非根本性的，可以在白盒模型中加以解释。但是，它们在未来的工作中仍然会存在。最后，需要进行过期性检测的客户端可以采用异步的方式进行，启用推测性的读取和补偿处理。[5,7]

7. 相关工作

本文研究基于若干相关领域：仲裁复制、一致性模型和量化一致性的方法。

在分布式系统和并行程序设计领域，如何管理复制的数据是一个被长期研究的问题。现在，有许多一致性模型可以解释人们在语义、性能和可用性之间的不同权衡情况。传统模型（比如可串行性和线性一致性）以及最近提出的模型（时间轴一致性 [8] 和并行快照隔离 [23]）均提供了“强”语义，但牺牲了高可用性，或者说，他们提供了一种能力，可以在所有副本上支持“始终启用”的响应行为。相比之下，面对高可用性和低延迟的需求，很多生产性的数据存储已经转向较弱的语义，以便在网络分区的情况中提供可用性。[9,26]

本文的重点放在现有广泛部署的系统所提供的语义上。由于实践中“强”一致性和最终一致性相当流行（而且 Dynamo 风格的系统需要明确选择这两种模型之一），我们把重点大致放在了这种二分法上。不过，也存在多种其他方式，但他们仍属于“弱”模型。例如，Bayou 系统提供了一系列的“会话保证”，包括读己之所写和单调读一致性。[25] 与此类似，德克萨斯州大学奥斯汀分校最近的技术报告声称，因果一致性的某个变体是在可实现的单向收敛（最终一致性）系统中可以达到最强的一致性模型，[18]。最近这个模型已经吸引了研究人员着手系统实现。[17] 正如我们上文中暗示的那样，概率的方法也可以用于本文未探讨的一致性模型。特别对于过期性而言，以前的研究，比如 TACT[27]，已经详细阐述了如何提供确定性的过期性边界。这种边界确定的过期性系统在确定性方面可与 PBS 媲美。

我们使用的用于分析现代部分仲裁系统的技术吸收了理论性较强的现有文献。在第 3 节中，我们简要地概述了仲裁复制 [20]。在本文中，我们特意利用了源于概率仲裁中的灵感来 [19] 分析扩展仲裁系统及其一致性。我们认为，在写入操作传播、反熵和 Dynamo 的背景下再次研究概率仲裁系统——包括非多数仲裁系

统（如树仲裁）——在理论工作方面前景广阔。我们研究了过期性的概率保证，而之前对于 *k*- 仲裁 [2, 3] 的研究已经检查了*确定性* 保证，即部分仲裁系统的返回值属于最近写入的 *k* 个版本。[2]

最后，最近的研究着重于从理论和实验两个角度测量和验证最终一致性系统的一致性（Rahman 等人提供了简要的概述 [21]）。这有助于我们验证一致性的预测并理解违反过期性的情况。

8. 结论

在本文中，我们介绍了 PBS，它可用于为最终一致的数据存储所返回的数据构建预期过期性模型。通过提供 SLA- 风格的一致性预测，PBS 为很多现有系统提供了另一种保证，可用于替代全有 - 或 - 全无的一致性保证。把先前的理论扩展到概率仲裁系统上后，我们推导出了分析性的解决方案用于分析部分仲裁系统的（*K*, *p*）- 正规语义，其中读取操作的预期过期性以版本的形式呈现。我们还分析了 Dynamo 风格的仲裁复制中的（Δ, *p*）- 正规语义，或者读取操作在实时方面的预期过期性。为了实现上述分析，我们开发了 ***WARS*** 延迟模型来解释 Dynamo 中的消息重排序如何导致过期性。为了仔细观察实践中（Δ, *p*）- 正规语义的延迟影响，我们使用了来自互联网公司的真实世界轨迹来驱动蒙特卡罗分析。我们发现，最终一致的 Dynamo 风格仲裁配置往往能在几十毫秒后达到一致，主要原因是他们能适应每台服务器的延迟方差。我们得出结论，在操作无故障时，最终一致的部分仲裁复制方案不仅经常返回一致的数据，还能提供相当大的延迟收益。我们认为，随着研究人员不断地研究和部署量化一致性度量，实用的终端用户功能将得以实现，同时之前模糊和饱受争议的复制配置也会变得清晰明了。

互动的演示

关于 Dynamo 风格的 PBS 的互动演示，请访问 http://pbs.cs.berkeley.edu/#demo。

鸣谢

本文大大受益于前文提到的众多个人的反馈。[5, 7] 同时，它还得到了下列公司的无偿支持：谷歌、SAP、亚马逊云计算服务、Blue Goji, Cloudera、爱立信、 通用电气、惠普、华为、IBM、英特尔、MarkLogic、微软、日本电气公司实验室、NetApp、NTT 多媒体通信实验室公司、Oracle、Quanta、Splunk 和 VMware。本文材料基于下列基金会的资助项目：美国国家科学基金会研究生助研奖学金（项目号 DGE 1106400）、美国国家科学基金会（资助号 IIS-0713661、 CNS-0722077 和 IIS-0803690）、NSF CISE 探 索（NSF CISE Expeditions）（项目号 CCF-1139158）、美国空军科学研究办公室（项目号 FA95500810352）和美国国防高级研究计划局（合同号 FA865011C7136）。

C

参考资料

1. Abadi, D.J.Consistency tradeoffs in modern distributed database system design:CAP is only part of the story. *IEEE Comput.45*, 2 (2012), 37–42.
2. Aiyer, A., Alvisi, L., Bazzi, R.A.On the availability of non-strict quorum systems.In *DISC 2005*.
3. Aiyer, A.S., Alvisi, L., Bazzi, R.A.Byzantine and multi-writer k-quorums.In *DISC* (2006), 443–458.
4. Bailis, P., Ghodsi, A. Eventual consistency today:Limitations, extensions, and beyond.*ACM Queue 11*, 3 (Mar. 2013), 20:20–20:32.
5. Bailis, P., Venkataraman, S., Franklin, M.J., Hellerstein, J.M., Stoica, I. Probabilistically bounded staleness for practical partial quorums.*PVLDB 5*, 8 (2012), 776–787.
6. Bailis, P., Venkataraman, S., Franklin, M.J., Hellerstein, J.M., Stoica, I. PBS at work:Advancing data management with consistency metrics.In *SIGMOD 2013 Demo*.
7. Bailis, P., Venkataraman, S., Franklin, M.J., Hellerstein, J.M., Stoica, I. Quantifying eventual consistency with PBS.*VLDB J.*(2014). (see http://link.springer.com/article/10.1007/s00778-013-0330-1).
8. Cooper, B.F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.A., Puz, N., Weaver, D., Yerneni, R. Pnuts:Yahoo!'s hosted data serving platform.In *VLDB 2008*.
9. Davidson, S., Garcia-Moina, H., Skeen, D. Consistency in partitioned networks.*ACM Comput.Surv.17*, 3 (1985), 314–370.
10. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W. Dynamo:Amazon's highly available key-value store.In *SOSP 2007*, 205–220.
11. Golab, W., Li, X., Shah, M.A.Analyzing consistency properties for fun and profit.In *PODC* (2011), 197–206.
12. Hamilton, J. Perspectives:I love eventual consistency but… http://perspectives.mvdirona.com/2010/02/24/ILoveEventualConsistencyBut.aspx (24 Feb. 2010).
13. Herlihy, M., Wing, J.M.Linearizability:A correctness condition for concurrent objects.*ACM Trans.Program.Lang.Syst.12*, 3 (1990), 463–492.
14. Kirkell, J. Consistency or bust:Breaking a Riak cluster. http://www.oscon.com/oscon2011/public/schedule/detail/19762.Talk at O'Reilly OSCON 2011 (27 Jul.2011).
15. Linden, G. Make data useful. https://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-29.ppt (29 Nov. 2006).
16. Linden, G. Marissa Mayer at Web 2.0. http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html (9 Nov. 2006).
17. Lloyd, W., Freedman, M.J., Kaminsky, M., Andersen, D.G.Stronger semantics for low-latency geo-replicated storage.In *NSDI 2013*.
18. Mahajan, P., Alvisi, L., Dahlin, M. *Consistency, Availability, Convergence*.Technical Report TR-11-22, Computer Science Department, University of Texas at Austin, 2011.
19. Malkhi, D., Reiter, M., Wool, A., Wright, R. Probabilistic quorum systems.*Inform.Commun.170* (2001), 184–206.
20. Merideth, M., Reiter, M. Selected results from the latest decade of quorum systems research. In *Replication*, B. Charron-Bost, F. Pedone, and A. Schiper, eds. Volume 5959 of *LNCS* (2010). Springer, 185–206.
21. Rahman, M., Golab, W., AuYoung, A., Keeton, K., Wylie, J. Toward a principled framework for benchmarking consistency.In *Proceedings of the 8th Workshop on Hot Topics in System Dependability* (Hollywood, CA, 2012), USENIX.
22. Schurman, E., Brutlag, J. Performance related changes and their user impact.Presented at *Velocity Web Performance and Operations Conference* (San Jose, CA, Jun.2009).
23. Sovran, Y., Power, R., Aguilera, M.K., Li, J. Transactional storage for geo-replicated systems.In *SOSP 2011*.
24. Stonebraker, M. Urban myths about SQL. http://voltdb.com/_pdf/VoltDB-MikeStonebraker-SQLMythsWebinar-060310.pdf. VoltDB Webinar (Jun.2010).
25. Terry, D.B., Demers, A.J., Petersen, K., Spreitzer, M.J., Theimer, M.M., Welch, B.B.Session guarantees for weakly consistent replicated data. In *PDIS 1994*.
26. Vogels, W. Eventually consistent. *CACM 52* (2009), 40–44.
27. Yu, H., Vahdat, A. Design and evaluation of a conit-based continuous consistency model for replicated services.*ACM Trans.Comput.Syst.20*, 3 (2002), 239–282.

Peter Bailis、Shivaram Venkataraman、Michael J. Franklin、Joseph M. Hellerstein 和 Ion Stoica（{pbailis, shivaram, franklin, hellerstein, istoica}@cs.berkeley.edu）来自加州大学伯克利分校。

译文责任编辑：陈海波