计算机科学 1996 Vol. 23 №. 1

我们应当重新考虑时间的问题

——93 图灵奖演讲*)

Richard Edwin Stearns 著

刘瑞挺 宋战江 译(南开大学计算机与系统科学系)

1993 年度 ACM 图灵奖由二位作者分享。本刊 1995 年№2 发表了 Juris Hartmanis 的讲演稿。他的不平凡经历和精辟而独到的见解,引起了广泛兴趣,今应读者要求,又将 R. E. Stearns 的讲演稿全文译出,以飨读者。这又是一个从数学家演变成计算机科学家的经历。讲到了(数学)对策论讨论竞争的本质与计算机科学讨论计算的本质的内在联系。文中对计算中的难度概念作了透彻分析,把确定性时间与难度证据之间的联系作了重要概括。这对深刻理解计算的本质很有参考价值。

在我念大学时,计算机科学还没有出现,我的兴趣严格说是数学。1958 年我从 Carlton 学院毕业并获得数学学士学位,接着就来到 Princeton 读数学研究生。在 Harold W. Kuhn 的领导和 Robert J. Aumann 的指导下,我完成了关于对策论的题为"无偏袒的三人合作博弈"的博士论文[8]并于 1961 年秋季毕业。

1960 年我受雇于 Richard L. Shuey,来到了位于纽约 Schenectady 的通用电气研究实验室(简称 GE)进行暑期工作。Shuey 是一个"信息研究小组"的负责人,该组包括 Juris Hartmanis 和 Philip M. Lewis I 在内,并且它是一个更大的名为"电子物理"的组织的一部分。整个夏季,我都和 Hartmanis 共同进行已由他开始的关于时序机分解的工作。结果我们写成了第一篇以 Hartmanis-Stearns 共同署名的文章[3]。GE 实验室成员的素质以及他们能自己来确定其研究目标的自由给我留下了深刻的印象,所以当 Shuey 在 1961 年邀请我返回 GE 作为一名终身雇员时,我为自己能获此良机而欣然接受。由于 GE 刚刚为与其它两个公司的合作定价问题调解了一起反垄断诉讼案,从而对我的博士论文开了一些玩笑(因为其中根本不涉及任何偏袒)。

当我在 GE 刚开始工作时,还只是作为一名数学家。研究实验室中没有计算机,我也从来没有使用过计算机。几年后,实验室得到了一台 GE300 计算机并安装上了一份在 Dartmouth 开发的分时系统。我的第一次编程实践是使用这台机器上的 BASIC 语言并通过它的电传机终端实现的。当 Hartmanis 和我开始研究计算复杂度理论之后,我们就开始进行计算机实践了。我转变成计算机科学家并不是一种有意识的决定而是一种演变,也就是当我在从事着我觉得最有兴趣的数学问题的研究过程中,竟发现自己已经置身于计算机科学之中了。

我继续保持着同时在对策论和计算机科学双方面的兴趣。实际上,我与 Hartmanis 共同完成的第一篇文章就是我随后论文工作的先导。你可能会问是否这些领域之间有一些共性的东西,我想是这样的。

一个共同点是他们都起源于 John von Neumann(冯·诺依曼)。对策论的产生很明显是由 von Neumann 和 Morgenstern 那本著名的书[13]《对策论与经济行为》为标志的。同时,von Neumann 由于提出了"von Neumann 存储程序的计算机模型"而被认为是计算机科学的奠基人。

第二个共同点是它们都需要弄清楚某些无形的却又很现实的东西。在对策论中,它是竞争的实质;而在计算机科学中,它是计算的实质。这两个需要全新数学方法的领域都要求确立新的数学模型。我在对策论方面的兴趣开始于本科生阶段的一个夏天。当时我在消遣中读到了上述著作[13]并看到了人们在对模型选择的讨论中倾注了多么大的注意力。

对我来说,最有趣的数学应该是那些告诉我们关于现实世界某些情况的数学问题。因此,作为数学家和计算机科学家我们应当提出的第一个问题是"我们的模型在多好的程度上反映了我们想描述的

*)本文由 R. E. Stearns 教授准许译成中文并在我刊发表。

对象或情况的显著特征?"结论的重要性更依赖于它所表达的信息而不是其证明的复杂性。象在软件中一样,"无用输入无用输出"同样适用于理论研究。

复杂度

我与 Hartmanis 共同进行的复杂度方面的工作发表在会议上[5]和杂志[6]中两种版本(还有研究实验室报告)。会议版本是在关于开关电路理论和逻辑设计的第五届年会上发表的,开关电路理论和逻辑设计这一对名字经历一些变化,现在则叫 FOCS(计算机科学基础)。文章的两种版本的标题中都首次使用了"计算复杂度"这个词,因此,我们是最早把这项工作命名为"计算复杂度"的。

尽管会议版本是在杂志版本之前发表的,但它却是晚于杂志版本而写的,并可称之为杂志版本的修订版。其中的一个修订版是对[1]所提到的 Blum 在 MIT 的博士论文的引用。他独立于我们所完成的工作,提供了一种对复杂度类更抽象的见解,这为促进复杂度领域早期的研究和迅速发展做出了很有意义的贡献。

其次,一个修订中我们利用较少输入并可生成 0/1 无限序列的多带图灵机模型提出了自己的理论[6]。从本质上来说,它与 Yamada 在"实时可计数函数"[11]方面的工作使用了相同的模型。然而,如今人们熟悉的语言识别模型在其所处的自动机理论中已迅速上升到了显著的位置。识别器是一种很有价值的模型,这是因为尽管它们只有简单的"是/否"输出,但这已足够用来讨论大多数可计算的问题,而且,它们对于非确定性的研究也是很理想的。在这一版本中,讨论了我们的方法在这个模型中的应用。

第三个修订版是讨论了使用 Hennie-Stearns 双带 $n\log n$ 模拟[7]而不使用 Hartmanis-Stearns 单带 n^2 模拟的前提条件(Fred Hennie 作为访问学者曾在 GE 工作),这个前提是区分复杂度类的更严格的充分条件。

我们工作的主要贡献是使用了确定性时间来定义复杂度类。利用现代的符号和语言识别模型,我们的定义可表示为:

定义 1 DTIME(T(n))是存在多带图灵机的所有语言 L 的集合,其中的图灵机能:

1. 回答"输入 w 是否属于语言 L?"的问题,并且

2. 最多经过 $T(|w|)$ 步就回答问题,其中 $|w|$ 是输入 w 的长度。

换句话说,对于一个问题,如果能够提供一个可在 T(n)步内回答其相应归属的问题(其中 n 是输入的大小),那么它就可归入 DTIME(T(n))类中。在算法分析中最普通的目标就是求出算法运行时间的上界,用时间复杂度模型来解释,这就相当于把此算法求解的问题归入一个复杂度类中。

从某种意义上说,这些复杂度类应当称作"容易度类",因为要说明一个问题是固有难解的(即要确定一个下界)就需要说明它并不属于某个时间类。

我们要先证明的一件事是"加速定理":

定理 1*) 对于所有的 $c>0$

$$DTIME(T(n))=DTIME(c\cdot T(n))。$$

换句话说,O(T(n))是重要的,而 T(n)却不是。这是我们应该从一个正确模型中所期待得到的性质,因为不存在什么有意义的方法来把自动机转换的次数与以秒为量度的时间之间建立起联系(由于以秒为单位的用来执行一个算法的时间将随着计算机技术的变化而不同)。把这种性质看作我们的定义的推论是很好的。当然在实践中常量因子也是很重要的,而且不是所有 O(T(n))的程序都同样良好,但是我们不想在自动机理论的层次上区分这些程序。

我们工作的核心结论是:

定理 2**) 如果 $\lim_{n\to\infty}\frac{T(n)\cdot\log(T(n))}{U(n)}=0$,

则 DTIME(U(n))包含一个不在 DTIME(T(n))中的语言。

这个结论表明了确实存在着一个时间的层次,使得时间函数的微小改变将会得出不同的类。例如,存在某些问题,对于所有的 $\epsilon>0$,它可以在 $O(n^2)$的时间内却不能在 $O(n^{2-\epsilon})$的时间内得到求解。因而,某些问题具有固有的复杂性从而不能被巧妙的编程所绕过。

我们的模型所不具有的一个性质是"机器不相关性"。即如果复杂度类是定义在某些增强的图灵机上(例如具有二维带的),那么这个类就会有些不同。我们的文章研究了若干种这类的增强情况,并发现时间上的差异是低阶多项式相关的。当复杂度的概念不随在时间上与图灵机多项式相关的模型而改变时,我们就认为它是"机器不相关"的。

不久以后,Hartmanis 和我与 Philip Lewis 共同研究了"带"的复杂度(现在称为空间复杂度)[10],它包含了在模型层次上的一个创新。我们根据所使用

---

*)为了叙述方便,定理的陈述稍有简化。 **)原先未用文[7],我们得出$[T(n)]^2$而不是 $T(n)\log T(n)$。同时 U(n)应该是"时间可构造的"。

的可读写纸带总量(即只读的输入带将不会被算在此总量之内)而定义了空间复杂度。这就可以使每次线性的复杂度类都能降到$\log n$的空间，甚至在某些情况下还可降到$\log\log n$的空间。

难度的概念

尽管时间与空间复杂度类形成了层次结构并提供了可用来讨论难度的概念，但是仍旧没有什么明显的技巧来判明一个问题是不容易求解的。换句话说，仍旧很难说明对于一个特定的问题根本就没有什么好方法可用来求解它。正如同假设我已有一个很聪明的方法可在$O(T(n))$的时间内求解一个问题，但是我怎么才能知道并不存在另外一个$O(T'(n))$的算法并且$T'(n)$比$T(n)$更小呢?

当Cook引入了我们现在称之为NP-难度与NP-完全性的概念[2]后这种情况才得到了很大的改观。其总体思想是把一个特定问题的难度与某些看上去困难的问题集合的难度建立起关联。这可如此实现：如果对于给定的问题存在着一个好算法，那么对此问题集内的每个问题也就都存在着一个好算法。按照Cook的说法，看上去困难的问题集就是现在被称为NP-完全的问题的集合，并且一个算法如果只需花费多项式的时间，那么它就被认为是"好的"。难度概念的本身现在被称为NP-完全性。

NP-完全的问题看起来似乎很难，这是因为从定义上来看，如果它们之间的任何一个可以在多项式的时间内求解，那么任何具有非确定多项式算法的问题也就都可以在多项式时间内求解。而实际上，这些问题似乎需要指数级的时间。用来证明一个问题X是NP-完全的标准方法就是要找到一个已知是NP-完全的问题Y并且要证明可以使用某些多项式时间的归约来把Y的任意实例转化为具有相同回答的X的一个实例。实际上，这个归约过程就说明了任何可用来求解X的方法都可用来同样有效地求解Y，所以X就不会比Y更容易。既然我们认为Y是难解的，那么我们也就认为X是难解的，所以NP-难度被认为是说明一个问题需要指数求解时间的很好的证据。

要实施这项计划，就有必要知道某些问题是NP-完全的。Cook的文章首先说明了一个被称作SAT的问题(即CNF布尔公式的可满足性问题)是NP-完全的。很快Karp就在其随后的工作中说明了许多有实际意义的组合问题也是NP-完全的[1]。不久就有几百个实际问题被证明是NP-完全的，其中许多问题在文[3]中出现。

不久以后又出现了另一个难度的概念，即PSPACE-难度，它是基于PSPACE-完全性概念的。PSPACE-难度问题看起来是很困难的，这是因为如果它们之中的任何一个可以在多项式时间内求解，那么在多项式空间内的所有问题就都可以在多项式时间内求解。PSPACE-完全问题看起来也需要指数级的时间。象NP-完全性一样，证明PSPACE-完全性的标准方法涉及来自于PSAPACE-完全问题的多项式时间归约。PSPACE-难度比NP-难度有更强的难度证据，这是因为即使NP-完全问题或许可以在多项式时间内求解，但PSPACE-难度问题却仍需要指数级求解时间。首先被证明是PSPACE-完全的问题之一为QSAT，即用来判定一个量化的CNF公式是否为真的问题[4]。

难度的概念已被证明为在对各种各样的实际问题进行分类方面是很有用的，并且为我们对计算的现实世界的理解作出了很大的贡献。这些思想的应用很成功，以致于我们有时忽略了它们的局限或是忘记了其真正的含义。在这里将指出它们的某些局限，然后我们可以看出，从确定性时间的观点来看，在这类问题的时间复杂度方面仍有很多值得探索之处。

虽然PSPACE-完全性比NP-完全性在难度方面有更强的证据，但是却没有理由认为PSPACE-完全问题需要更多的求解时间从而更难。例如我们可以考虑NP-完全的SAT问题和PSPACE-完全的QSAT问题。已知的对这些问题的最好算法在实际上是等价的，它们都涉及对变量的所有$\Theta(2^n)$赋值的考虑和使用$\Theta(n)$的空间。我们倾向于认为SAT不可能更快地求解，因而我们也就倾向于认为SAT和QSAT都具有相同的复杂度。

当我们说NP-完全问题似乎占用"指数级时间"时，我们不是指的2^n，在这种情况下，"指数"意味着$2^{n^a}(a>0)$，所以2^{n^2}、$2^{\sqrt{n}}$、$2^{\sqrt[3]{n}}$，以及甚至是$2^{\sqrt{n}\log n}$等都是指数。实际上，类$DTIME(2^{n^\varepsilon})$包含了PSPACE-完全和NP-完全问题(对于所有的$\varepsilon>0$)。当ε较小时，时间$\Theta(2^{n^\varepsilon})$不是不可能很大。例如，当$n=31,991$时，在一台1MIPS的计算机上$2^{\sqrt{n}}$次的运算需要1小时来完成，而$2^{\sqrt{n}}$次的运算只需大约1061秒。

因此可以想象，某些这种"困难"的问题对于所有大尺度的输入都可以很容易地得到解决。

一个基于时间的观点

为了便于进一步的讨论，考虑以下由Harry B. Hunt Ⅲ和我提出的基于时间的幂指数概念[4]：

定义2　问题L的幂指数定义为：

当集合$\{r|L\in DTIME(2^{n^r})\}$不为空时，它是此

集合的最大下界；

当此集合为空时，它是∞。

这个概念有几个令人感兴趣的性质：

1. 由于每个非负实数的非空集合都有一个最大的下界，所以每个问题都有一个幂指数；

2. 对于每个有理数 r，都存在着一个具有幂指数 r 的问题（这可以从定理 2 推出）；

3. 幂指数的概念是"机器不相关"的。

具有多项式算法的问题（即在 P 中的问题）都具有幂指数 0。具有指数时间界限的问题（即在 EXPTIME 中的问题）都具有有限的幂指数。

如果任何一个 NP-完全的问题具有大于 0 的幂指数，那么所有的 NP-完全问题也就都这样并且使得 P≠NP；对于 PSPACE-完全问题也是如此。因此若能证明某个 NP-完全问题具有非零的幂指数也就证明了 P≠NP，而若能证明某个 PSPACE-完全问题具有非零的幂指数则也就证明了 P≠PSPACE。因此我们预计在确定这些问题的幂指数方面将会遇到很多困难。然而，如同对于"难度的证据"一样，我们可以使用归约的方法来研究"幂指数的证据"。为此，我们必须注意"归约的大小"。它可以定义如下：

定义 3 一个归约 R 的大小为 s(n)，当且仅当输出 R(w)的长度为 $\Theta(s(n))$。

幂指数之间的关系可以如下确定：

定理 3 如果 L_1 的幂指数是 r，并且 L_1 可以通过大小为 n^s 的归约在多项式时间内归约为 L_2，那么 L_2 的幂指数至少是 r/s。

注意，归约的尺寸越大（即度 s 越大），下界 r/s 就越小。若仅知道一个归约是多项式的，那么就根本不会为幂指数提供任何信息。

为了理解定理 3，我们可以考虑 SAT 和 CLIQUE 问题。从 SAT 到 CLIQUE 的标准归约[5-1]具有 n^2 的大小并且不知道有更好的归约方法，因此从此定理中可得到的最强结论是 CLIQUE 的幂指数至少为 SAT 幂指数的一半。这确实很好地符合已知的事实，即对 SAT 已知的最好的算法需花费 $2^{O(n)}$ 时间，而对 CLIQUE 的最好算法只有使用 $2^{O(\sqrt{n})}$ 的时间。如果我们能够找到一个从 SAT 到 CLIQUE 的线性大小的归约，那么我们就可以为 SAT 得到一个 $2^{O(\sqrt{n})}$ 时间的算法！

从确定性时间的角度来看，并不是所有的 NP-完全问题都具有同样的难度。紧紧围绕着归约的大小，我们可以对各种 NP-完全问题的复杂度进行更加深刻的比较。一种评估我们的理解的方法是提出如下问题：假设 SAT 确实需要 $2^{\Theta(n)}$ 的时间（即假定 SAT 的幂指数是 1），那么在对其它 NP-完全问题的幂指数方面可以推出什么结论？

在许多情况下，我们可以从 SAT 到其它在 $2^{O(n)}$ 时间内可解的问题进行线性的归约，并且这一问题的回答是它们的幂指数也都必须为 1。然而，仍有许多问题具有已知的非线性的归约，而这些问题本身却可能更容易求解。在大多数情况下，是否这种可能性能设法实现或是否有更小的归约存在，这是一个未解决的问题。

在更加理论化的层次上，就我们所能知道的，幂指数使我们得出结论：时间的层次和与难度证据相关的类之间是极其不相关的。我们一个最好的猜测是世界看起来与图 1 中所画出的相似。这张图表明 EXPTIME 的集合被分割成具有相等幂指数的类，这些类也同样分割了 PSPACE 和 NP 的集合。从图中看出，SAT 和 QSAT 都是和幂指数 1 相关联的。在它们的下面，CLIQUE 是与幂指数 1/2 相关联的，而 P 则包含在具有为 0 幂指数的问题之内。

我们把确定性时间与以难度证据相联系起来的观点可以概括如下：

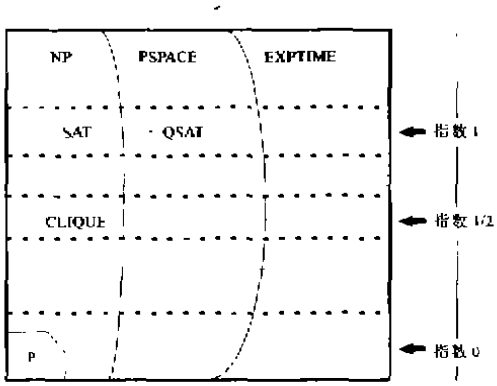

图 1 被幂指数所分割的 EXPSPACE

- 所有的 NP-完全问题并不是都具有相同的难度；
- 所有的 PSPACE-完全问题也不是都具有相同的难度；
- PSPACE-完全的问题可以比 NP-完全的问题更易求解；
- 即使 SAT 需要 $2^{\Theta(n)}$ 时间，那么也存在着这种可能性，即许多实际中的 NP-完全问题会只需要少得多的时间去求解。

参考文献 见《Comm. of the ACM》，Nov. 1994