关于计算问题的已求解实例的生成

Martín Abadi*

Eric Allender†

Andrei Broder*

Joan Feigenbaum†

Lane A. Hemachandra§

摘要：我们考虑计算问题的已求解实例的高效生成。特别地，我们考虑无懈可击的生成器。设 S 是 $\{0,1\}^*$ 的一个子集，M 是接受 S 的图灵机；M 在输入 x 上的一个接受计算 w 被称为 $x \in S$ 的一个“证言”。非正式地说，一个程序是 α-无懈可击生成器，如果它在输入 1^n 时，按照某个分布生成实例-证言对 $\langle x, w \rangle$，其中 $|x| = n$，使得任何多项式时间的敌手在给定 x 的情况下，对于无穷多个长度 n，至少以概率 α 无法找到 $x \in S$ 的一个证言。

哪些集合具有无懈可击生成器的问题在理论上具有内在的吸引力，其结果可以应用于启发式算法的测试数据生成以及零知识证明系统理论。无懈可击生成器的存在与密码学安全单向函数的存在密切相关。我们证明了三个关于无懈可击性的定理。第一个定理处理了如果确实有 NP 集合具有无懈可击生成器，那么哪些 NP 集合有的问题。第二个定理处理了这些生成器的无懈可击程度。

定理（完全性）：如果 NP 中有任何一个集合具有 α-无懈可击生成器，那么 SAT 也有一个。

定理（放大）：如果 $S \in \mathbf{NP}$ 有一个 β-无懈可击生成器，其中 $\beta \in (0,1)$ 是某个常数，那么对于每个常数 $\alpha \in (0,1)$，S 都有一个 α-无懈可击生成器。

我们关于无懈可击性的第三个定理表明，无法使用可相对化的技术来解决单凭 $\mathbf{P} \neq \mathbf{NP}$ 的假设是否足以证明无懈可击生成器存在的问题。显然，存在一些相对化世界，其中存在无懈可击生成器；在所有这些世界中，$\mathbf{P} \neq \mathbf{NP}$。我们第三个定理解决的更微妙的问题是，是否也存在 $\mathbf{P} \neq \mathbf{NP}$ 但不存在无懈可击生成器的相对化世界。

定理（相对化）：存在一个谕示，相对于它 $\mathbf{P} \neq \mathbf{NP}$，但不存在无懈可击生成器。

1 引言

Sanchis 和 Fulk 研究了困难问题的测试实例的复杂性，以及此类构造与复杂性类结构之间的联系 [20,21]。在本文中，我们考虑计算问题的已求解实例的高效生成。例如，如果 $S = \{x: \exists w.p(x,w)\}$ 是 NP 中的一个集合，我们可能希望按照指定的分布生成实例-证言对 $\langle x, w \rangle$。生成对 $\langle x, w \rangle$ 的复杂性与给定 x 找到 w 的

复杂性之间的关系在理论上具有内在趣味性，并且对于测试困难问题的启发式算法以及零知识证明系统的应用提案也很重要。

具体来说，我们询问是否可能生成我们所谓的实例-证言对的无懈可击分布。例如，是否可能生成对 $\langle f, a \rangle$，其中 f 是一个布尔公式，a 是一个满足赋值，将秘密 a 给用户 A，公布公式 f，并合理地确信多项式时间的敌手将无法找到 f 的满足赋值 a'，从而冒充 A？Feige、Fiat 和 Shamir 提出将“身份的零知识证明”的这种用途作为一种安全机制；他们建议的具体方案基于二次剩余问题（QRP, [6]）。即使 QRP 结果比广泛认为的要容易，身份的零知识证明可能仍然有用；此外，即使 QRP 是困难的，也有可能基于另一个问题构建方案并获得更高的安全性。因此，拥有一个复杂性理论框架来考虑生成实例-证言对的方案是否产生安全分布是很重要的。

当 Goldwasser、Micali 和 Rackoff 首次引入零知识证明系统时，他们假设了一个全能的证明者（[10]）。此后，他们和其他人（例如 [3], [5]）考虑了一个模型，其中证明者和验证者拥有相同的计算资源，证明者唯一的优势在于他恰好知道手头困难问题的特定实例 x 的证言 w，也许是因为他同时构造了 x 和 w。这个模型，连同所有 NP 集合都有零知识证明系统的证明（[4], [11]），构成了将多方协议“编译”成“已验证”协议的基础（[7], [11]）。因此，重要的是要认识到，只有当存在一种高效程序可以生成比验证者能解决的更困难的实例时，这个模型才有意义。

许多 NP 完全集合有明显的简单生成方案。例如，n 个顶点上的哈密顿图可以通过选择一个随机回路，然后以 $1/2$ 的概率独立地添加每个其他可能的边来生成。生成特定图的概率与其拥有的哈密顿回路的数量成正比。然而，以下例子表明，一些生成已求解实例的自然方法并不安全。第一种方法被一个非常简单的算法攻破；第二种方法可以被一种复杂的技术破解。

示例：3SAT。一个 3SAT 实例由变量集 $U = \{u_1, u_2, \ldots, u_n\}$ 和子句集 $C = \{c_1, c_2, \ldots, c_m\}$ 组成，其中每个子句由三个文字组成。问题是是否存在满足 C 的真值赋值。（定义见 [8]。）

生成已求解 3SAT 实例的一种“自然”方法如下。从 2^{n} 种可能性中均匀选择一个真值赋值 t。对于 1 到 m 之间的每个 i，均匀随机选择三个不同的变量；对应于这些变量的八组文字中，有七组在 t 下为真。从那七组中均匀随机选择子句 c_{i}。该方案以相等的概率生成 t 满足的每个由 m 个子句组成的集合 C。

如果子句数量 m 足够大，多项式时间的敌手可以以高概率重构 t。基本的观察是，如果 $t(u_i) = \text{TRUE}$，那么

因此，如果 $m \geq kn \ln n$，其中 k 是一个合适的常数，那么以概率 $1 - o(1)$ 对每个 i 同时，文字 u_i 在 C 中比文字 $\overline{u}_i$ 出现得更频繁，当且仅当 $t(u_i) = \text{TRUE}$。

人们可以尝试改进这个生成方案，通过选择每个子句中的文字使得至少有一个为 FALSE，至少有一个为 TRUE。那么对于所有 i，$\overline{u_i}$ 的期望出现次数等于 u_i 的期望出现次数。然而，如果 $m \geq kn^2 \ln n$，通过观察关于变量对的统计量，改进后的方案很容易被破解。

示例：子集和。一个子集和实例由一个正整数有限集 $A = \{a_1, a_2, \ldots, a_n\}$ 和一个正整数 M 组成。问题是是否存在子集 $A' \subset A$，其和等于 M。子集和问题的难度是背包型公钥密码系统的依据。

可以如下生成已求解的子集和实例。均匀随机选择由零和一组成的向量 $e = (e_1, \ldots, e_n)$。固定一个正整数 B。将每个 $a_i \in A$ 从 $\{1, 2, \ldots, B\}$ 中均匀随机选择。令 $M = \sum_{1 \leq i \leq n} a_i e_i$。

这个生成方案可以被 Lagarias 和 Odlyzko 的算法破解（[16]）。如果 B 足够大，那么每个实例几乎肯定可以通过他们巧妙应用 LLL 基约减算法来求解。

在下文的第 3 节中，我们将精确定义生成方案是无懈可击的含义。然后我们证明一个完全性定理，该定理指出，如果任何 NP 集合有一个无懈可击生成器，那么 SAT 也有一个。特别地，在二次剩余假设、离散对数假设或因数分解假设下，可以生成 SAT 的硬分布。[1] 这并不奇怪。更有趣的是，即使所有这些假设结果都是错误的，只要可以生成 NP 中任何东西的硬分布，仍然可以生成 SAT 的硬分布。我们为 SAT 构建的无懈可击生成器结合了任何可能的 NP 集合生成器中存在的任何无懈可击性，并且不假设它知道无懈可击性来自何处（如果它通过乘以不同的素数来构建困难实例，如同在 [6] 中那样，就会做出这种假设）。第 3 节还包含一个放大定理，展示了如何增强任何可生成分布的无懈可击性，以及一个相对化定理——无懈可击生成器的存在显然意味着 $\mathbf{P} \neq \mathbf{NP}$，但其逆命题无法通过可相对化的技术来证明。

在第 4 节中，我们简要讨论了哪些集合可以按照哪些分布生成的一般性问题，考虑了几项相关工作，并提出了未来研究的方向。第 2 节包含了本文其余部分广泛使用的术语和符号。为了节省空间，我们将完整的证明推迟到论文的最终版本；只要可能，我们会给出传达一些要点的概要。

2 术语、符号和约定

我们称一个抛硬币并在所有输入上在最坏情况多项式时间内终止的程序为随机多项式时间程序。令 $\{M_i\}$ 表示随机多项式时间程序的标准枚举。令 $\{N_j\}$ 表示多项式时间非确定性程序的标准枚举；因此，每个 NP 集合至少被我们枚举中的一个程序识别。我们用 $L(N_j)$ 表示 N_j 识别的集合（或语言）。

设 N 是一个非确定性多项式时间程序，S 是 $L(N)$。我们称 N 在输入 x 上的每个接受路径为实例 x 在 S 中的一个证言。不失一般性，我们假设对于任何固定的程序 N，实例的长度 n 决定了证言的长度 m，并且函数 $n \mapsto n + m$ 是单射。我们用 ω_{x}^{N} 表示 $x \in S$ 的证言集合。

我们用 PF 表示多项式时间可计算函数类；函数 $f \in \mathbf{PF}$ 的值域不必是 $\{0,1\}$，因此 PF 是计算 $\mathbf{P}$ 中集合成员资格的函数集的真超集。

我们用 S_{n} 表示长度为 n 的 S 的元素。符号 Λ 表示程序的默认输出；它可以用来指示所需的输出不存在或程序未能找到它。我们考虑的所有生成程序都以一元形式输入长度 n，在多项式时间内运行，并产生 S_{n} 的元素；因此，根据定义，我们将注意力限制在高效生成上。

3 无懈可击的生成器

在本节中，我们提供了一个复杂性理论框架来考虑困难、已求解实例的生成。我们精确定义了实例-证言对的分布“对于多项式时间敌手是安全的”意味着什么。我们的第一个定理解决了如果确实有这样的集合，那么 NP 中哪些集合有无懈可击生成器的问题。定理 2 解决了这些生成器究竟有多无懈可击的问题。最后，定理 3 解决了证明无懈可击生成器存在需要什么样的复杂性理论假设的问题。

定义：第 $(i,j)^{\mathrm{th}}$ 个生成方案，记为 $G_{i,j}$，是一个程序，在输入 1^{n} 时，首先模拟输入 1^{n} 上的 M_{i} 并获得输出字符串 y。如果 y 的形式是 $\langle x,w\rangle$，其中 $|x| = n$ 且 w 是 N_{j} 在输入 x 上的接受计算，那么 $G_{i,j}$ 输出 $\langle x,w\rangle$；否则，输出 Λ。

考虑以下游戏，在生成方案 $G_{i,j}$ 和 PF 中的敌手 f 之间进行。游戏的输入是一个字符串 1^n；第一步是运行输入 1^n 上的 $G_{i,j}$。如果 $G_{i,j}$ 输出一个对 $\langle x, w \rangle$，那么第二步是 f 输出 $f(x)$；如果 $G_{i,j}$ 输出 Λ，那么游戏在第一步后结束。如果生成器输出 Λ，或者生成器输出 $\langle x, w \rangle$ 并且 $f(x)$ 是 N_j 在输入 x 上的接受计算 w'，则函数 f 赢得游戏；否则，生成器获胜。注意 w' 不必等于 w；例如，在第 1 节的识别方案中，如果敌手 f 计算出用户 A 的公共公式的任何满足赋值，他就可以危及用户 A 的安全性——他不需要发现在密钥分发期间给 A 的私有赋值。

定义：一个生成方案是 α-无懈可击的，其中 α 是 $[0,1]$ 中的一个常数，如果对于所有 $f \in \mathbf{PF}$，存在无穷多个长度 n，使得 f 在输入 1^n 上获胜的概率至多为 $1 - \alpha$。该概率是在输入 1^n 上多次运行游戏计算得到的。

定义：NP 中的一个集合 S 是 α-无懈可击的，其中 α 是 [0,1] 中的一个常数，如果存在一对 (i,j)，使得 $G_{i,j}$ 是 α-无懈可击的并且 $S = L(N_j)$。

注意，无懈可击生成器与密码学安全单向函数密切相关。设 g 是 PF 中一个保长函数，并假设任何多项式时间程序在无穷多个长度上至少无法反转 g 输出的一个常数部分（其中“反转”意味着“找到原像中的某个元素”）。那么 g 的像有一个无懈可击的生成方案：在输入 1^n 上，生成一个长度为 n 的随机 w，并令 x 等于 $g(w)$。类似地，一个无懈可击的生成方案 $G_{i,j}$ 产生一个密码学安全单向函数。程序 M_i 可以被视为从抛硬币序列到对 $\langle x, w \rangle$ 的映射。设 g 是将抛硬币序列映射到 M_i 输出的对的首分量 x 的函数。那么 g 对于任何多项式时间的敌手在无穷多个长度上必须难以反转；如果不是，敌手可以发现产生 $\langle x, w \rangle$ 的抛硬币序列，方案 $G_{i,j}$ 将是不安全的。如果我们要求在两种情况下敌手在所有足够大的长度上都失败，而不仅仅是在无穷多个长度上，同样的评论也适用。

我们并不声称根据我们的定义无懈可击的生成方案在实践中必然有用。例如，[6] 中的密钥分发者当然希望知道的不只是存在无穷多个长度使得特定的多项式有界敌手可以以高概率被阻止；他还希望知道这些长度是实用规模的，并拥有找到它们的程序。然而，我们的无懈可击性定义确实为开始复杂性理论研究提供了一个很好的起点。

定理 1（完全性）：如果任何 NP 集合对于某个正的 α 是 α-无懈可击的，那么 SAT 也是 α-无懈可击的。

证明（概要）：完整的证明分三个阶段进行。首先，我们构造一个“通用生成方案” G_U，它模拟所有可能的生成方案，捕获其中任何一个方案中存在的无懈可击性的一个常数部分。接下来，我们为 SAT 构造一个生成器，它以一种保持无懈可击性的方式将 Cook 归约应用于 G_U 生成的集合 S_U。最后，我们表明损失的那部分无懈可击性可以被重新捕获。

对于通用生成器 G_U，我们需要一个程序 M_U，其运行时间由一个特定的多项式界定，来模拟无限多个程序，这些程序的单个运行时间可能是任意高次的多项式。我们通过以下引理克服这个障碍；它保证我们只需要考虑生成器 $\{G_k\}$，其中程序 M 在平方时间内运行。

引理：如果 $G_{i,j}$ 是 α-无懈可击的，那么存在一个 α-无懈可击的生成方案 $G_{i',j'}$，其中 $M_{i'}$ 在平方时间内运行。

我们不能使用像 Cook 证明 SAT 是 NP 完全性时所用的“泛型归约”来构造通用生成器。这样的归约不一定是长度一致的（即，将相同长度的实例映射到相同长度的实例）。此外，即使我们的泛型归约将长度为 n 的实例映射到长度为 n^k 的实例，它也可能不保持无懈可击性：非正式地说，如果特定生成器 G_m 输出的“困难实例”在长度 n 上代表概率质量的一个常数部分 α，那么它们的像不一定代表长度 n^k 上概率质量的一个常数部分，仅仅是因为长度为 n^k 的实例要多得多。

我们使用一个非标准的配对函数来克服这个困难。它将正整数划分如下“列”：第 m 列由所有形如 $2^{m-1} + k \cdot 2^m$ 的整数组成，其中 $k \geq 0$。每个输入长度 n 恰好落入一列——其索引比 n 的二进制表示中最低有效“1”位的索引多一。在输入 1^n 上，G_U 首先找到 m，即包含 n 的列的索引，然后从区间 $[n - 2^m, n)$ 中均匀选择一个整数 l。接着，G_U 在输入 1^l 上模拟 G_m 得到 $\langle x, w \rangle$，填充 x，并输出 $\langle x10^{n-l-1}, w \rangle$。

引理：如果 G_{m} 是 α-无懈可击的，那么 G_{U} 是 $(\alpha / 2^{m})$-无懈可击的。

非正式地说，为了证明对于 PF 中的所有 f，存在无穷多个长度 n，使得 f 以至少 $\alpha / 2^m$ 的概率未能“破解” G_U 的输出，我们证明任何这样的 f 对应于一个函数 f'，它在无穷多个长度 n' 上以至少 α 的概率未能破解 G_m 的输出。损失因子 2^m 的发生是因为 f' 和 G_m 的“硬长度” n' 对应于 f 和 G_U 的硬长度 n，使得 $n' \in [n - 2^m, n)$；因此 G_U 仅以概率 2^{-m} 选择在输入 $1^{n'}$ 上模拟 G_m。（注意 $\alpha / 2^m$ 确实是一个常数，因为 m 只是我们枚举 $\{G_k\}$ 中一个生成器的（固定）索引。）

为了构造一个 $(\alpha / 2^{m})$-无懈可击的 SAT 生成器 G_{SAT}，我们利用生成器 G_U 中的程序 M_U 在立方时间内运行这一事实。我们修改 Cook 归约，使得当应用于在立方时间内运行的 NP 机器时，它取长度为 k 的实例并产生长度恰好为 k^4 的实例。这个修改的 Cook 归约 r 也引出了一个从 S_U 成员资格的证言到 SAT 元素的满足赋值的映射。因此，G_{SAT} 在输入 1^n 上的行为如下。如果 n 不是完全四次幂，它输出 Λ。否则，它在输入 1^k 上模拟 G_U，其中 $k^4 = n$，获得一个对 $\langle x, w \rangle$，并输出 $r(\langle x, w \rangle)$。我们在完整论文中证明 G_{SAT} 至少和 G_U 一样无懈可击。

下面的定理 2 保证，如果 SAT 有一个 $(\alpha / 2^m)$-无懈可击生成器，那么它也有一个 α-无懈可击生成器。

推论：在二次剩余假设、离散对数假设或因数分解假设下，对于某个 $\alpha \in (0,1)$，存在一个 SAT 的 α-无懈可击生成器。

备注 1：出于密码学目的，人们真正想要的不仅仅是“存在一个无限的硬长度集合”。注意，定理 1 的证明给出了一些希望，因为如果某个 G_{m} 在 1 和 n 之间的 $t(n)$ 个长度上击败了一个敌手，那么 G_{U} 在 1 和 n 之间的 $\Omega(t(n))$ 个长度上击败了相应的敌手。（如果我们使用一个将其两个参数都二次拉伸的标准配对函数，这就不成立了。）

定理 2（放大）：如果一个 NP 集合 S 对于某个正的 β 是 β-无懈可击的，那么对于所有 $\alpha \in (0,1)$，S 也是 α-无懈可击的。

证明（概要）：只需证明 α-无懈可击性意味着 $2\alpha / (1 + \alpha)$-无懈可击性即可，因为由 $\alpha_0 = \alpha$, $\alpha_i = 2\alpha_{i-1} / (1 + \alpha_{i-1})$ 定义的序列的极限是 1。

直观地说，我们将展示以最自然的方式提高无懈可击性水平：生成实例，尝试破解它们，并丢弃被破解的。假设 $G_{i,j}$ 是 α-无懈可击的且 $S = L(N_j)$。如果 $G_{i,j}$ 是 $(\alpha + (1 - \alpha) / 2)$-无懈可击的，那么我们就完成了，因为 $(\alpha + (1 - \alpha) / 2) > (2\alpha / (1 + \alpha))$；所以假设它不是。那么，根据定义，存在某个 $f \in \mathbb{P}\mathbb{F}$，它在除了有限多个输入 1^n 外，以大于 $(1 - \alpha) / 2$ 的概率赢得与 $G_{i,j}$ 的对抗。

考虑生成器 $G_{i',j}$，它在输入 1^n 上的工作方式如下：首先它像 $G_{i,j}$ 一样在输入 1^n 上运行 M_i。如果 M_i 输出 $\langle x, w \rangle$，那么 $G_{i',j}$ 计算 $f(x)$ 并检查它是否是 N_j 在输入 x 上的接受计算。如果是，那么 $G_{i',j}$ 再次在输入 1^n 上运行 M_i；否则，$G_{i',j}$ 输出 $\langle x, w \rangle$。如果 f 连续赢得足够多轮游戏，那么 $G_{i',j}$ 输出 Λ。

显然，$G_{i',j}$ 生成与 $G_{i,j}$ 相同的集合，即 $L(N_j)$。在完整论文中，我们证明 $G_{i',j}$ 是 $(2\alpha/(1+\alpha))$-无懈可击的，并推导出 $G_{i',j}$ 必须模拟的 f 与 $G_{i,j}$ 之间游戏轮数的足够好的界限。

备注 2：为简单起见，我们将敌手建模为确定性的多项式时间函数。显然，在实践中，人们必须防范随机化的多项式时间敌手。定理 1 和定理 2 所述即使我们在无懈可击性的定义中对所有随机化多项式时间函数进行量化也成立。我们在完整论文中提供细节。

在定理 1 中，是否有可能弱化至少一个 NP 集合是 α-无懈可击的假设？显然存在一些相对化谕示，相对于它们存在无懈可击生成器。确实，一个随机谕示就能做到（[19]）。在所有这些相对化世界中，$\mathbf{P} \neq \mathbf{NP}$。$\mathbf{P} \neq \mathbf{NP}$ 的假设是否足以证明无懈可击生成器的存在？我们的下一个定理表明，这样的证明将是不可相对化的。

定理 3（相对化）：存在一个谕示 B，使得 $\mathbf{P}^{\mathbf{B}}\neq \mathbf{N}\mathbf{P}^{\mathbf{B}}$，并且相对于 B 不存在无懈可击生成器。

证明（概要）：令 $\mathbf{B} = \mathbf{QBF} \oplus K$，其中 $\oplus$ 是不相交并，K 是一个极其稀疏的字符串集合，具有最大的柯尔莫哥洛夫复杂性。具体来说，K 包含每个长度 n_i 的一个字符串，其中序列 $n_1, n_2, \ldots$ 定义如下：$n_1 = 2$，n_i 是 n_{i-1} 的三重指数，对于 $i > 1$；如果 $x \in K$ 且 $|x| = n$，那么 x 的柯尔莫哥洛夫复杂度为 n。包含 QBF 使得能访问 $\mathbf{B}$ 的机器拥有 PSPACE 的全部能力。

使用 [13] 中的技术，直接证明 $\mathbf{P}^{\mathbf{B}} \neq \mathbf{N}\mathbf{P}^{\mathbf{B}}$ 是简单的。

为了证明相对于 $\mathbf{B}$ 不存在无懈可击生成器，设 $G_{i,j}$ 是一个可以访问谕示的生成方案，并假设它是 α-无懈可击的，其中 α 是 $(0,1)$ 中的某个常数。我们通过构造一个 $\mathbf{PF}^{\mathbf{B}}$ 中的敌手 f 来导出矛盾，该敌手能够破解任何长度的超过 $1 - \alpha$ 比例的实例。以下是 f 的非正式描述及其工作原理：

生成器 $G_{i,j}$ 涉及一个随机化多项式时间程序 M_i 和一个非确定性多项式时间程序 N_j，两者都可以在任何步骤查询 B。令 n^{k_1} 和 n^{k_2} 分别是 M_i 和 N_j 运行时间的界限，令 k 是一个大于 $\max(k_1, k_2)$ 的整数。当试图破解长度为 n 的实例 x 时，f 首先构造集合 K'，包含所有长度小于 $\log(n^{ck})$ 的 K 的元素，其中 c 是适当选择的常数。因为 K 非常稀疏，所以 $K \setminus K'$ 中最多有一个字符串 r，$G_{i,j}$ 在生成长度为 n 的 x 时可能查询过 B 关于它。

假设 N 是最接近 n 的整数，使得存在一个长度为 N 的 K 中的字符串。困难的情况是当 $\log(N^{ck}) \leq n \leq 2^{N / ck}$ 时；否则，f 可以通过查询 $\mathbf{B}' = \mathbf{QBF} \oplus K'$ 来构造 $x \in L(N_j)$ 的证言。所以假设，例如，$n = 2^{N / ck}$。

破解者 f 首先使用 $\mathbf{B}'$ 来确定是否存在一个抛硬币序列 s，如果 $G_{i,j}$ 使用 $\mathbf{B}'$，它将导致 $G_{i,j}$ 在输入 1^n 上输出 x。如果这样的 s 存在，那么 f 可以使用 PSPACE 构造一个，并进而构造一个证言；这个构造可能涉及也可能不涉及发现随机字符串 $r \in K \setminus K'$。如果这样的 s 不存在，那么 f 无法构造证言。然而，我们证明，只有当 $G_{i,j}$ 实际查询了 $\mathbf{B}$ 关于 r 在 K 中的成员资格时，才没有这样的 s（因此也才只有那时 f 失败）。我们用一个计数论证来完成证明，该论证表明，如果这以任何常数概率 α 发生，那么 r 不可能具有最大的柯尔莫哥洛夫复杂度。

4 讨论、相关工作与开放问题

设 S 是一个 NP 集合，并固定一个接受 S 的特定机器 N。回想 ω_{x}^{N} 是 N 在输入 x 上的接受路径集合。如果存在一个随机化多项式时间程序，在输入 1^{n} 时生成对 $\langle x, u \rangle$，其中 $|x| = n$，使得赋予 x 的概率与 $|\omega_{x}^{N}|$ 成正比，则称 S 是**规范生成**的。第 1 节中给出的哈密顿图、3SAT 公式和子集和实例的简单生成方案都是规范的，就这些集合通常的证言类型而言。

我们称这些生成器为规范的主要是因为，在某种意义上，NP 集合的所有生成器都是规范的。如果 $G_{i,j}$ 是 S 的一个生成方案，那么导致 M_i 输出 $\langle x, w \rangle$ 的抛硬币序列就是 $x \in S$ 的一个证言，赋予特定 x 的概率显然与导致它被输出的抛硬币序列的数量成正比。

哈密顿图的简单规范生成方案具有这种一般形式：均匀生成 w，然后从所有使得 w 是 x 在 S 中的证言的实例集合中均匀选取 x。事实上，许多 NP 集合（例如 SAT、具有完美匹配的图、具有大小为 $|V(G)| / 2$ 的团的图）就通常的证言类型而言都具有这种形式的规范生成方案。这自然引出了一个问题：是否每个 NP 集合就每种证言类型而言都有这样的规范生成方案？这个问题的答案是否定的，除非 NP 集合的构造问题总是可以在多项式时间内解决。（构造问题是：给定一个实例 x，如果 x 是 yes-实例，则找到一个证言，如果 x 是 no-实例，则说明没有证言。）

一个有趣的研究领域是可生成（即规范）分布与 Levin 等人研究的“平均困难”分布之间的关系（[17]，另见 [12], [15] 以及更近期的 [22]）。Levin 的随机化 NP（记为 RNP）是一类对 (D, μ)，其中 D 是 NP 中的任何决策问题，μ 是 $\{0, 1\}^*$（解释为 D 的实例）上的任何概率函数，其累积分布函数 $\mu^{*}(x) = \Sigma_{z < x}\mu(z)$ 是多项式时间可计算的。在 [22] 中，Venkatesan 和 Levin 将定义扩展到 NP 中的构造问题；他们允许的分布仍然是那些具有多项式时间可计算的 μ^{*} 的分布。

Venkatesan 和 Levin 展示了一个是 RNP-困难的构造问题，即，如果存在一个算法可以在期望多项式时间内解决它，那么所有 RNP 构造问题都可以在期望多项式时间内解决。他们考虑的实例分布易于生成；然而，它赋予 no-实例正概率。

这提出了一些自然的问题。是否存在一个 RNP-困难分布（一个构造问题的实例分布）仅赋予 yes-实例正概率？如果坚持随实例一起生成证言，能否高效生成该分布？分布是高效可生成的与它具有高效可计算的 μ^{*} 这两个要求是否互斥？例如，我们针对哈密顿图的规范生成方案产生的分布可能不具有多项式时间可计算的 μ^{*}：如果它有，那么计算图中哈密顿回路数量的 #P 完全问题将在多项式时间内可解。

最后，我们想提及，已求解实例的生成也由 Rardin、Tovey 和 Pilcher [18] 考虑过；他们的目标是构建启发式算法的测试实例。

5 致谢

我们感谢 Mike Foster、Steve Mahaney、Steven Rudich 和 Mihalis Yannakakis 的有益讨论。我们特别感谢 Laura Sanchis。

参考文献

[1] M. Blum and S. Micali. "How to Generate Cryptographically Strong Sequences of Pseudo-random Bits," SIAM J. on Comput. (13), 1984, 850-864.

[2] R. Boppana and R. Hirschfeld. "Pseudorandom Generators and Complexity Classes," to appear in Advances in Computer Research, Silvio Micali (ed.), JAI Press (pub.), 1987.

[3] G. Brassard and C. Crépeau. “Non-transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond,” Proceedings of the 27^{th} FOCS, IEEE, 1986, 188-195.

[4] G. Brassard and C. Crépeau. “Zero-Knowledge Simulation of Boolean Circuits,” Advances in Cryptology — CRYPTO86 Proceedings, Andrew Odlyzko (ed.), Springer-Verlag (pub.), 1987, 223–233.

[5] G. Brassard, D. Chaum, and C. Crépeau. "Minimum Disclosure Proofs of Knowledge," to appear.

[6] U. Feige, A. Fiat, and A. Shamir. “Zero Knowledge Proofs of Identity,” Proceedings of the 19^{th} STOC, ACM, 1987, 210-217.

[7] Z. Galil, S. Haber, and M. Yung. "Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model," Advances in Cryptology — CRYPTO87 Proceedings, Carl Pomerance (ed.), Springer-Verlag (pub.), 1988, 135–155.

[8] M. Garey and D. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[9] S. Goldwasser and S. Micali. “Probabilistic Encryption,” JCSS (28), 1984, 270-299.

[10] S. Goldwasser, S. Micali, and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems," to appear in SIAM J. on Comput.

[11] O. Goldreich, S. Micali, and A. Wigderson. "Proofs that Yield Nothing but their Validity and a Method of Cryptographic Protocol Design," Proceedings of the 27^{th} FOCS, IEEE, 1986, 174-187.

[12] Y. Gurevich. “Complete and Incomplete Randomized NP Problems,” Proceedings of the 28th FOCS, IEEE, 1987, 111-117.

[13] J. Hartmanis. "Generalized Kolmogorov Complexity and the Structure of Feasible Computations," Proceedings of the 24^{th} FOCS, IEEE, 1983, 439-445.

[14] M. Jerrum, L. Valiant, and V. Vazirani. "Random Generation of Combinatorial Structures from a Uniform Distribution," TCS (43), 1986, 169-188.

[15] D. Johnson. “The NP-Completeness Column, An Ongoing Guide,” JOA (5), 1984, 284–299.

[16] J. Lagarias and A. Oldlyzko. "Solving Low-Density Subset Sum Problems," JACM (32), 1985, 229-246.

[17] L. Levin. “Average Case Complete Problems,” SIAM J. on Comput. (15), 1986, 285-286.

[18] R. Rardin, C. Tovey, and M. Pilcher. "Polynomial Constructability and Traveling Salesman Problems of Intermediate Complexity," ONR-URI Computational Combinatorics Report CC-88-2, Purdue University, November, 1988.

[19] S. Rudich, private communication.

[20] L. Sanchis and M. Fulk. "Efficient Language Instance Generation", University of Rochester Computer Science Department TR 235, 1988.

[21] L. Sanchis. “Test Instance Construction for NP-hard Problems,” University of Rochester Computer Science Department TR 206, 1987.

[22] R. Venkatesan and L. Levin. "Random Instances of a Graph Coloring Problem are Hard," Proceedings of the 20^{th} STOC, ACM, 1988, 217-222.

[23] A. C. Yao. “Theory and Applications of Trapdoor Functions,” Proceedings of the 23^{rd} FOCS, IEEE, 1982, 80-91.