美国数学学会期刊 第00卷，第0期，000–000页 S
0894-0347(XX)0000-0

arXiv:2303.03958v1 [cs.CC] 6 Mar 2023

P 与 NP的线性相关性

BOJIN ZHENG 和 王伟武

1. Introduction the P and NP

如今， $P\ vs\ NP$问题或 $P?=NP$是计算机科学中最重要的问题。Stephen Cook 于1971年提出了这一问题，当时他发表了论文《定理证明过程的复杂性》 [2]。2000年，这一问题被选为七个千年大奖难题之一。数十年来，大量研究人员开发了各种技术，试图解决这一问题[1], ，但至今仍未完全解决。

通俗地说， P 指可由确定性图灵机（DTM 或 TM）在多项式时间内求解的问题集合，而 NP 指可由确定性图灵机在多项式时间内验证的问题集合。 $P\ vs\ NP$ 问题或 $P?=NP$ 问题的含义是：任何能够被高效验证的问题，是否都能够被高效求解。Cook 提出了 NP–完全性 [2]，并描述了 NP中最难的 NP–完全问题。 NP–完全问题可以归约为 $3SAT$问题，也就是说，如果一个 NP–完全问题能够被高效求解，那么所有 NP–完全问题都能够被高效求解。目前，实践中已发现数千个 NP–完全问题，涉及生物学、化学、经济学等数十个学科。解决 $P?=NP$ 问题不仅具有重要的实际意义，也具有重要的科学和哲学意义。

2. Basic Strategy to Solve P vs NP

NP的定义基于非确定性图灵机（NDTM），它可被视为一种可预见的多项式时间算法。因为 $P\subseteq NP$、$P\neq NP$实际上意味着 $P\subset NP$，或者说， NP中至少有一个元素不属于 P，即无法由一般的、不可预见的多项式时间算法解决。

一些研究者坚持认为，任何关于证明 $P\neq NP$ 的论述都应证明，至少有一个 NP 不能由任何不可预见的多项式时间算法解决，或者说，不能事先由一般的、不可预见的多项式时间算法解决。实际上，这种信念不仅可疑，而且没有必要。

当我们谈到“一个通用、无法预见的多项式时间算法”时，在这里我们将它称为“神话算法”，这实际上意味着我们已经假定它存在。至少，我们赋予了它一个名称或短语，也就是说，它在语法意义上存在。当然，它在形式上也存在。从形式逻辑的角度来看，我们

编辑收到日期：2023年2月22日。
2010 *Mathematics Subject Classification.* 主要。

©XXXX 美国数学学会

无法证明它不存在。然而，我们的任务一直是证明神话算法不存在。由此便产生了逻辑矛盾。

从根本上说，在证明 $P \neq NP$ 之前，我们无法证明神话算法不存在。这里的论证并非我们的发明。历史上，亚里士多德曾以类似的论证讨论真空不存在的问题。伽罗瓦也面临同样的困境，并找到了一种方法，证明五次方程不存在一般代数方法。根据伽罗瓦的例子，没有必要讨论神话算法。顺便一提，“神话算法”这一说法的存在，已经预设了 $P = NP$，因此不能用来证明 $P \neq NP$。

证明不存在的一种常见方法，是基于存在的特征或性质。例如，如果我们有 $\forall x \in X, f(x) = 1$ and $\exists y, f(y) = 0$，显然就可以得到 $y \notin X$。也就是说，如果我们有 $\forall p \in P, LinearDep(p) = true$，并且$\exists np \in NP, LinearDep(np) = false$，那么就能得到 $np \notin P$，即 $P \neq NP$。

事实上，预测 $LinearDep(x)$ 意味着 x 的线性相关。也就是说，如果我们能证明，对于 P 的任意实例，它都是线性相关的，而对于 NP 的一个实例，它是线性无关的，那么我们就证明了 $P \neq NP$。注意到$3SAT \in NP$，我们的证明将归结为两个命题：

(1) $\forall p \in P, LinearDep(p) = true($

2) $\exists 3sat, LinearDep(3sat) = false$

当然，这两个命题也同样可行：

(1) $\forall p \in P, LinearDep(p) = true(2)$

$\exists np \in NP, LinearDep(np) = false$

3. Definitions and Basic Concepts

3.1. 可计算问题的离散表示。

定义 3.1. 可计算问题的离散表示。对于任意给定的可计算问题，都可以将其表示为输入和输出对的集合，即 $< X, Y >$。

定理 3.2. 任意可计算问题与其时间最优算法之间存在一一映射关系。

证明。首先证明，任意可计算问题都必然对应一个时间最优算法。对于任意可计算问题，都存在一系列算法来解决该问题。对于任意算法，都有一个求解时间度量。由于该度量值构成全序关系，因此必然存在一个时间最优算法。接着证明，任意时间最优算法都必然对应一个可计算问题。对于该时间最优算法，其输入和输出写作一个元组 $< X, Y >$，这必然对应原问题。因此，可计算问题与时间最优算法之间存在一一映射关系，也就是等价关系。□

关于定理3.2，Oded Goldreich 谈到了相同的内容[3]：“.....在计算理论中，对象的表示扮演着核心角色。从某种意义上说，计算只是将一个对象的一种表示转换为同一对象的另一种表示。.....事实上，任何完全明确的问题的答案都隐含在问题本身之中，而计算则被用来使这一答案显现出来。”

根据定理 3.2，本文中， P问题的概念与多项式确定性算法相同，即 P 算法（记作 P_f）；而 NP问题的概念与 NP 问题的确定性算法相同。

3.2. 计算与表示的哲学。根据定理3.2和 Oded 的观点，我们可以将问题和算法视为一枚硬币的两面。至于这枚硬币，我们称之为“系统”。也就是说，问题是系统的一种表示，而算法是另一种表示。我们将系统表示为一个元组 $< X, f(x)\ Y >$。显然，这个元组可以表示为一个矩阵，其各行表示该元组的记录或实例，我们称这个矩阵为导出矩阵。

（时间最优）算法、问题和系统可以视为同一事物，并且它们共享同一个导出矩阵。在这里，我们将导出矩阵称为系统的离散表示，它可以改写为一个元组 $< X, f(x)\ Y >$，即系统表示。

由于算法等同于系统和问题，因此可以将其视为算法表示。

考虑到任何算法都可以理解为图灵机，相应的图灵机称为图灵机表示。如果存在等价的抽象机，相应的机器则可称为机器表示。例如，随机存取机。

由于通用图灵机的存在，图灵机可以编码为一个整数（程序代码），即一个数，因此我们可以用整数来表示问题。也就是说，问题具有数域表示。

由于图灵机的执行过程可以建模为一个图，其中节点表示状态，因此所有执行过程都可以组织成一棵树，根节点是停机状态，叶节点是初始状态。这种方法称为拓扑表示。

当然，SAT问题可以被视为一种基于符号逻辑的表示，我们可以称之为SAT表示或符号逻辑表示。

当我们从表示的角度思考 P和 NP的本质时，会产生一些有趣的问题。例如，将数域表示扩展到复数形式会发生什么？如何扩展随机存取机，使其能够在多项式时间内处理 NP？ P和 NP的拓扑图有何区别？如何扩展系统表示来表示 NP? 等等。本文不会回答所有这些问题，其中一些将留待未来研究。

3.3. 元组的线性相关性。假设自然数 S可以表示为一个符号串 $S = S_n \cdots S_2 S_1$，其中 $S_i \in \{0, 1\}$，则 S可以视为一个维度为 n的位向量。本文采用列向量，因为列向量便于与自然数的表示相对应。

考虑元组 $\Phi =< X, Y >$，令 $X = S = S_n \cdots S_2 S_1$, $Y \in \{0, 1\}$,$< X, Y >$构成一个新的（$n+1$）维向量 $\Phi_A = (S_n, \cdots, S_2, S_1, Y)$。

$\Phi \subseteq X \times Y$。此处， $\times$ 表示笛卡尔积。

当 Φ 线性相关且 X 线性无关时， Φ的维数将等于 n。

由于我们将讨论 P和 NP，它们的输入是自然数； X的维数（即自然数的二进制表示）会随 n而变化，因此我们将 X 视为一个整体，并令其维数为1。 Y也是如此，其维数为1。我们只关注 X 和 Y之间的线性相关性。

定义3.3。元组的线性相关性。如果向量集 Φ_A 线性相关，那么元组 Φ 线性相关。如果向量集 Φ_A 线性无关，那么元组 Φ 线性无关。

根据定义3.3，显然，如果 $< X$、$Y >$的向量集线性相关，那么 $< X$、$Y >$ 的维数为1，记作$|| < X$、$Y > || = 1$。如果线性无关，那么元组$< X$、$Y >$的维数为2，记作 $|| < X$、$Y > || = 2$。

定理3.4。假设有两个集合 A 和 B， X 和 Y 分别线性无关， A、 $B \subseteq X \times Y$和 $A \cap B = \emptyset$。令 $||A|| = 1$、$||B|| = 2$，则$||A \cup B|| = 2$。令 $||A|| = 1$、 $||B|| = 1$，则 $||A \cup B||$ 不确定，可能为1，也可能为2。令 $||A|| = 2$、 $||B|| = 2$，则 $||A \cup B|| = 2$。

定理3.4表明，两个线性相关元组的并集可能线性相关，也可能线性无关；两个线性无关元组的并集必定线性无关；如果一个元组线性相关，另一个元组线性无关，那么这两个元组的并集必定线性无关。

反过来，在线性无关的元组中可以找到线性相关的真子集；在线性相关的元组中无法找到线性无关的真子集，只能找到线性相关的真子集。

3.4 元组线性相关性示例。与计算机数据库中的术语类似，我们将元组的实例称为记录。

当记录集 $\{< x_0, y_0 >\}$ 只有一个元素时，它总是线性相关的。此处， $x_0 \in X$ 和 $y_0 \in Y$。

当 $< x_0$、$y_0 >=< 1$、$1 >$、 $k_1 = 1$、 $k_2 = -1$具有 $k_1 * 1 + k_2 * (-1) = 0$时，因此它是线性相关的。

记录集 $\{< 0, 0 >, < 1, 1 >\}$ 是线性相关的。

对于记录集 $\{< 1, 0 >, < 0, 1 >\}$，可以计算得出该集合是线性无关的。因此，$\{< 1, 0 >, < 0, 1 >, < 0, 0 >\}$ 和 $\{< 1, 0 >, <0$、$1 >$、< 0、$0 >$、< 1、$1 >\}$ 是线性无关的。

对于记录集 $\{< 1, 0 >, < 0, 1 >\}$，如果 $y' = \bar{y}$，那么我们得到 $\{< 1, 1 >, < 0, 0 >\}$，此时它是线性相关的。这一事实表明，在某些情况下，向量元素的逆元可能会导致其线性相关性发生变化。

但是 $\{< 1, 0 >, < 0, 1 >, < 1, 1 >\}$ 是线性无关的，经过逆变换得到的结果 $\{< 1, 1 >, < 0, 0 >, < 1, 0 >\}$ 也线性无关。

3.5. 系统的线性相关性。该系统可以写作 $<X, f, Y >$。这里， f： $X \to Y, X = \mathbb{N}$, $Y = \{0, 1\}$；因为 f可以表示为一台图灵机，其代码是一个二进制常量。由于 f 有结果， f 通常可以改写为 $< f_{code}, f(x) >$。也就是说， $< X, f, Y >$也可以改写为

$< X, f_{code}, f(x)\ Y >$。 f_{code}可以转换为某个通用图灵机下的程序。

3.6. 系统的导出矩阵的定义。一般来说， P_f 可以表示为 P_f： $X \to Y$, $X = x_1x_2\cdots x_n \subseteq \mathbb{N}, x_i \in \{0,1\}, Y = \{0,1\}, i \in \{1,n\}, Time(P_f) \in Poly(n)$。

P_f的导出矩阵，或者说系统 $< X, P_f, P_f(x)\ Y >$的导出矩阵，可以定义为

$$\Phi(P_f) = (x_1, x_2, \cdots, x_n, P_f, P_f(x), Y) \tag{3.1}$$

这里， P_f 是程序代码，即一个常数。

3.7. 输入中常数的影响。本文关注的是线性相关性。对于导出矩阵的线性相关性，一个显而易见的问题是：当输入中含有常数时会怎样?

实际上，在定义元组的线性相关性时，我们关注的是 Y与 $P_f(x)$之间的关系。当 $Y = P_f(x)$时，则线性相关；当$Y \neq P_f(x)$时，我们应根据线性空间的定义检查其线性无关性。但是，当对于某些 x， $Y \neq P_f(x)$ 和 $Y = P_f(x)$时，该系统将是线性无关的。

至于导出矩阵中的常数 P_f ，我们应忽略它，不要计算它与其他项的线性相关性。

3.8. 系统线性相关定理。因为 $f(x) = Y$,$< X, f_{code}, f(x)\ Y >$将线性相关。事实上， Y总是等于 $f(x)$，因此该系统总是线性相关。

显然， P_f（ $\forall p \in P$的函数）将有一个相应的线性相关系统，因为 P_f 是一个 f。因此，我们有 $LinearDep(p) = true$;

定理 3.5。 $\forall P_f, LinearDep(P_f) = true$

证明。略。 □

4. The Details of Proving $P \neq NP$

根据定理 3.5，任意 P 都是线性相关的。如果我们能找到一个线性无关的NP示例，就可以证明 $P \neq NP$。我们将阐明 NP的性质，并证明某些 NP是线性无关的。

4.1. 将系统扩展到复合布尔系统。

定义 4.1. 复布尔数。我们把表达式为 $z = a + bi, a \in \{0,1\}, b \in \{0,1\}$ 的数称为复布尔变量，其实部为 $\theta(z) = a$。 $0 + 0 * i = 0$。

定义 4.2. 复布尔变量的值集。复布尔数的值集称为复布尔集。记作 $CB = \{0, i, 1, 1+i\}$。

定义 4.3。复布尔数的模。布尔数 $z = a + bi$ 的模 $\psi(z)$ 具有 z ,

$$\psi(z) = a \vee b \tag{4.1}$$

根据系统的定义，显然，输入基于实整数，而实整数可以视为布尔变量。众所周知，每个数都可以用 4 作为进位基数来表示。也就是说，我们可以将每 2 个比特组合成一个整体，这 2 个比特可以视为复布尔变量，其中一个比特是实部，另一个比特是虚部。

对于 $f(x)$ 的值，我们只关注实部，因此可以用 $\theta(f(x))$ 替换之前的 $f(x)$。对于 Y，因为 $Y\ inCB$，也就是说，它是一个复布尔变量，我们可以用 $\psi(Y)$ 替换之前的 Y。因此，复布尔系统可以改写为 $< X, f, \theta(f(x))\ \psi(Y) >$。

4.2. NP是复布尔系统。Cook 提出了 $NDTM$ 来描述 NP。在此，我们证明任何可由 $NDTM$ 识别的语言 L 都可以转换为关于复布尔变量的 CNF 公式。为了便于说明，我们给出图灵机和非确定性图灵机的定义。

定义 4.4（图灵机）。图灵机 M 是一个元组 $\langle \Sigma, \Gamma, Q, \delta \rangle$。$\Sigma$ 是一个包含输入符号的有限非空集合；Γ 是一个有限非空集合，其中包含空白符号 b 和 Σ；Q 是可能状态的集合，q_0 是初始状态，q_{accept} 是接受状态，q_{reject} 是拒绝状态；δ 是一个转移函数，满足

$$\delta : (Q - \{q_{accept}, q_{reject}\}) \times \Gamma \to Q \times \Gamma \times \{-1, 1\}$$

如果 $\delta(q, s) = (q', s', h)$，其含义是：如果 M处于状态 q并正在扫描符号s，那么 q' 是新状态，s' 是写入的符号，而读写头则根据 h 是 -1 还是 1，向左或向右移动一格。C 是 M的一个构型，$C = xqy,\ x, y \in \Gamma^*$,$y$ 非空，$q \in Q$。M 的计算是唯一的序列 $C_1, C_2, \cdots$。如果计算是有限的，那么步数就是构型的数量。

定义 4.5（非确定性图灵机）。非确定性图灵机 N是一个元组 $\langle \Sigma, \Gamma, Q, \delta_1, \delta_2 \rangle$。$N$ 与 M之间的区别在于转移函数。N中有两个函数 δ_1 和 δ_2。在任意构型C_i中，都可以在不同的函数之间进行选择。选择记为 $\Delta_{i,\ j}$,$j \in \{1, 2\}$。选择序列是一条计算路径。如果存在一条路径$\Delta_{1,\ j_1}, \Delta_{2,\ j_2}, \cdots$通向接受状态，那么 N停机并接受。计算次数是最短路径中的步数。

有一个引理可以计算 CNF [1]的大小。

引理 4.6。对于给定的布尔函数 $f\colon \{0,1\}^l \to \{0,1\}$，存在一个 CNF公式 φ，其大小为 $l2^l$，满足 $\varphi(u) = f(u)$，$\forall u \in \{0,1\}^l$。一个 CNF 公式的大小是 $\wedge$ 和 $\vee$的数量。

定理 4.7。如果语言 L 被非确定性图灵机N接受，那么 L可约化为复合布尔式 z的一个 CNF公式 φ_z。

证明。假设非确定性图灵机 N 在时间 $T(n)$ 内接受语言 L。存在一条路径 $\Delta_{1,\ j_1}$, $\Delta_{2,\ j_2}, \cdots, \Delta_{T(n)\ j}$，通向接受状态。

假设 $\delta'_0 = \delta_1 \cap \delta_2$, $\delta'_1 = \delta_1 - \delta'_0$, $\delta'_2 = \delta_2 - \delta'_0$，路径 $\Delta_{1,j_1}, \Delta_{2,j_2}, \cdots$变为路径 $\Delta'_{1,j'_1}, \Delta'_{2,j'_2}, \cdots$，其中 $\Delta'_{i,0}$是确定性的，$\Delta'_{i,1}$和 $\Delta'_{i,2}$是非确定性的。

例如，路径 $\Delta'_{1,0}, \Delta'_{2,1}, \Delta'_{3,2}, \Delta'_{4,0}, \cdots$， $\Delta'_{1,0}$ 和 $\Delta'_{4,0}$ 是确定性的，二者均来自 δ'_0; $\Delta'_{2,1}$ 和 $\Delta'_{3,2}$ 是非确定性的，分别来自 δ'_1 和δ'_2。

给定输入 $x \in \{0,1\}^n$，我们将构造一个命题公式 φ_z，使得φ_z 可满足，当且仅当 N 接受 x。

步骤 1：构造公式 F_1，以断言输入 x。假设 y_1、y_2、$\cdots$、y_i、$\cdots$、y_n、$y_i = x_i$。F_1 的大小为 $4n$。变量的数量为 n。F_1 出现在 z的 a 中。

步骤 2：构造公式 F_2，以断言初始配置 C_0。一个构型可以编码为长度为 c 的比特串 w_0。c 由 $|Q|$ 和 $|\Gamma|$决定。F_2 的大小为 $c2^c$。变量的数量为 c。F_2 也出现在 z的 a 中。

步骤 3：构造公式 F_3，以断言 $1 \le i \le T$（n）个构型。C_i（由w_i编码）由 C_{i-1}的状态、输入符号 $x_{inputpos(i)}$、带符号（由读写头 $C_{prev(i)}$的最后一次操作决定）决定。存在一个函数F（由 δ'_0、δ'_1 和 δ'_2决定），满足

$$C_i = F(C_{i-1}, x_{inputpos(i)}, C_{prev(i)})$$

对于每个构型 C_i，变量的数量至多为 $3c+1$。公式的大小为 $(3c+1)2^{(3c+1)}$。对于所有构型，公式的大小至多为 $T(n)(3c+1)2^{(3c+1)}$。

如果 C_i 在 $\Delta'_{i,j} = \Delta'_{i,0}$ 的路径上，那么 F 由 δ'_0 决定。F_3 存在于 z 的 a 中。
如果 C_i 在 $\Delta'_{i,j} = \Delta'_{i,1}$ 或 $\Delta'_{i,2}$ 的路径上，那么 F 由 δ'_1 或 δ'_2 决定。F_3 存在于 z 的 b 中。

步骤 4 构造公式 F_4，用于断言最终状态和输出。这与步骤 2 类似。

最后，我们将公式的所有部分进行 AND 运算，得到 $z = \varphi_z$。
最后，该公式断言 N 到达接受状态，当且仅当 φ_z 的 $a = 1$ 和 φ_z 的 $b = 1$。

□

φ_z 的构造是 N的一条计算路径。因此，可以通过N计算 φ_z。

4.3. 复数形式的导出矩阵。NP_f的导出矩阵，或者说，系统 $< X, NP_f, \theta(NP_f(x)) \ \psi(Y) >$的导出矩阵，可定义为

$$\Phi(NP_f) = (a_1, b_1, a_2, b_2, \cdots, a_n, b_n, NP_f, \theta(NP_f(x)), \psi(Y)) \tag{4.2}$$

这里，NP_f 是一段程序代码，即一个常数。

显然，当将 X视为实整数时，$\Phi(NP_f) = \Phi(P_f)$。也就是说，P的导出矩阵是 NP的一种特殊情况。

由于我们不关心 $\Phi()$中作为常数的 NP_f 和 P_f，因此可以将导出矩阵改写为

$$\Phi'(NP_f) = (a_1, b_1, a_2, b_2, \cdots, a_n, b_n, \theta(NP_f(x)), \psi(Y)) \tag{4.3}$$

4.4. 导出矩阵的线性无关性。

定理 4.8。
$\exists np \in NP, LinearDep(np) = false$

证明。假设 $x_0, np_f(x_0) = a_0 + i * b_0$。当 $b_0 = 0$, $\theta(np_f(x)) = \psi(a_0 + i * b_0)$。当 $b_0 = 1$, $a_0 = 0$, $\theta(np_f(x)) = 0 \neq \psi(a_0 + i * b_0) = 1$。因此，$LinearDep(np) = false$。□

定理 4.9. $P \neq NP$。

证明。根据定理 3.5 和定理 4.8， $P \neq NP$。 □

5. An Example and its Explanation

我们将用一个 3SAT 示例来说明 $P \neq NP$ 的验证。

5.1. SAT问题的定义。SAT问题也称为布尔可满足性问题。这个问题可以表述为：给定一个合取范式（合取形式的子句的一系列析取形式），询问是否存在一组赋值，使整个表达式为真。

如果存在满足条件的赋值，则该表达式可满足；如果不存在这样的赋值，则该表达式不可满足。

显然，如果我们希望整个合取范式为真，就需要每个子句都为真。而每个子句内部都是析取形式，因此只要至少有一个文字为真，该子句就为真。

形式上，这个问题可以表述为：

假设存在一个布尔向量 $x = (x_1, x_2, \cdots, x_n)$，以及一个合取范式 f，$f(x) = C_1 \wedge C_2 \cdots \wedge C_m$，其中，$C_l = \vee_1^k (x_j)$ $1 \leq l \leq m, 1 \leq j \leq n$，如果存在 x 使得 $f(x) = 1$，则输出 x；否则，输出 0。也就是说，结果集为 $y = (y_1, y_2)$，其中 $y_1 = x, y_2 = f(x)$。

当 $k = 2$ 时，它称为 $2SAT$ 问题；当 $k = 3$ 时，它称为$3SAT$ 问题。

5.2. 用 Tarjan 算法求解 $2SAT$ 问题。Tarjan 算法是一种非常高效的算法，可用于求解 $2SAT$ 问题。

Tarjan 算法的第一步是构建图。方法如下：分别设置 $2n$ 节点、 x_1、 x_2、 $\cdots$、 x_n 和 $\bar{x}_1$、 $\bar{x}_2$、 $\cdots$、 $\bar{x}_n$。然后将每个子句转换为 $u \to v$的形式，在图中画一条 $u \to v$ 线，并根据逆定理连接$\bar{v} \to \bar{u}$ 。

Tarjan 算法的第二步是确定强连通分量。

在有向图 G中，如果存在一条从 u 到 v的有向路径，同时也存在一条从 v 到 u的有向路径，则称这两个顶点强连通。若有向图 G 中的任意两个顶点都强连通，则称 G 为强连通图。

当对应于 x_i 的正文字和负文字的任意节点出现在一条强连通路径上时，该公式就是一个矛盾。因为在强连通路径上， $x_i \to \bar{x}_i$ 和 $\bar{x}_i \to x_i$ 构成一个环。因此， $\bar{x}_i$ 和 x_i 同时出现在该环中，这就构成了矛盾。

强连通图是一个线性相关向量。因为强连通图中的任意两个节点都是强连通的，根据强连通的定义， $u \to v$ 和 $v \to u$ 存在，也就是说， $u \equiv v$ 和 v 互为充分必要条件。因此， $u \equiv v$ ，也就是说，它们线性相关。

对于强连通图，可以进行收缩，也就是将强连通图中的所有变量替换为一个变量。

有向非强连通图中的极大强连通子图称为强连通分量。也就是说，在非强连通图中找出一个极大强连通子图；或者说，在不存在矛盾时，寻找线性无关向量。这个线性无关向量称为强连通分量。

Tarjan 算法的第三步是赋值。

赋值实际上是求解一个线性布尔方程。布尔方程很容易求解。

使用 Tarjan 算法排除线性无关的表达式 f 后，必定有，

$$k_1 * x_1 + k_2 * x_2 + \cdots + k_n * x_n - k_{n+1} * f = 1 \tag{5.1}$$

设 $q(x) = k_1 * x_1 + k_2 * x2 + \cdots + k_n * x_n$，则有，$q \in [0, 2]$。

如果 $q = 0$，那么 $x_1 = x_2 = \cdots = x_n = 0$。

如果 $q = 1$，那么选择一个 $i \in [1, n]$，令 $x_i = 1$，其余比特设为 0，并构造 x，由此该 x 有效，因此只需进行 n 验证。

如果 $q = 2$，那么选择 $i \in [1, n]$ 和 $j \in [1, n]$，$i \neq j$，令 $x_i = 1$ 和 $x_j = 1$，其余比特设为 0，并构造 x，由此该 x 有效，因此只需进行 $n*(n-1)/2$ 验证。

5.3. 关于 SAT问题和线性相关的解释。本节将解释 $2SAT$问题与 $3SAT$ 问题在线性相关性方面的本质区别。本质区别源于三条公理之间的冲突，或是二值逻辑中排中律的废弃。如果 $P = NP$，也就是说，若 $2SAT$ 的算法可用于解决 $3SAT$ 问题，那么它必然会导致代数和计算中的三条重要公理发生冲突，或者废弃二值逻辑中的排中律。

5.4. 计算与代数中的三条公理。在计算机科学中，有一条基本公理：计算步骤按顺序逐步执行。在代数中，有两条公理，即加法交换律和加法结合律。现表述如下。

定义 5.1. 顺序执行公理。在任何可信计算模型中，算法的执行步骤都满足全序关系。

定义 5.2. 加法交换律。对于加法，始终有 $a+b = b+a$。

定义 5.3. 加法结合律。对于加法，始终有 $(a+b)+c = a+(b+c)$。

显然，逻辑或运算 $\vee$ 是一种加法运算。

对于顺序执行公理，在并行计算场景中，某些算法的指令执行步骤存在一种等价的偏序关系，但本质上仍然是全序关系。确定性图灵机满足这一公理。

为便于讨论，这里只提及加法的交换律和结合律，而省略乘法的交换律和结合律。事实上，由于计算顺序执行公理

本质上是自然提出的，加法和乘法在交换律和结合律下并不相同。

5.5. $2SAT$ 问题的线性相关性。 $2SAT$问题中的 CNF 子句至多包含两个变量；如果存在二元运算符，则至多包含一个二元运算符。也就是说，在 $2SAT$ 问题中，逻辑或对 $\vee$ 进行操作时不满足结合律。对于合取范式，要求每个子句都为真。因此，在逻辑与操作下，每个子句的交换律和结合律与计算的顺序指令执行公理并不冲突。

根据离散表示， $2SAT$ 可以写成 $<< X$、 $f >Y >$，其中 $< X\, f >$是 $2SAT$的输入。也就是说，

$$2SAT =<< X, f >, Y > \tag{5.2}$$

对于任意可计算的 f， f可以写成 $< X\, Y >$。因此，存在，

$$2SAT =<< X, < X, Y >>, Y >=< X, Y > \tag{5.3}$$

由于输入中的 Y等于输出中的 Y，因此 $2SAT$ 问题必定线性相关。

5.6. $3SAT$ 问题的线性无关性。用相同的方法可以证明$2SAT$ 的线性相关性，似乎 $3SAT$ 也线性相关。然而，对于 $3SAT$ 问题，由于子句包含三个变量，因此有两个逻辑或运算，所以交换律和结合律会起作用。同时，由于存在多个逻辑或运算，计算的顺序执行公理也会起作用。这三条公理彼此冲突。

默认情况下，计算模型按照从左到右的顺序对各个子句求值。
在 SAT问题中， CNF公式的不可满足性映射为 $y = 0, y \in Y$。

在 Tarjan 算法中， CNF 公式的不可满足性被映射为这样一个事实：比特变量的真值和假值不能同时处于同一个强连通环中，也就是说，不存在一个同时包含某个变量真值和假值的环。

我们基于 $2SAT$ 问题构造一个简单的 $3SAT$ 问题。其 CNF公式为：

$$f(x_1, x_2, x_3) = x_1 \wedge x_2 \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \tag{5.4}$$

根据结合律，使用 Tarjan 算法计算第三个子句 $\bar{x}_1 \vee \bar{x}_2 \vee x_3$， $\bar{x}_1 \vee \bar{x}_2 \vee x_3 =$ ($\bar{x}_1 \vee \bar{x}_2$) $\vee\, x_3$。首先计算 $\bar{x}_1 \vee \bar{x}_2$，得到未满足值。此时，Tarjan 算法不会读取 $\vee x_3$。

根据交换律和结合律， $\bar{x}_1 \vee \bar{x}_2 \vee x_3 =$ ($\bar{x}_1 \vee x_3$)$\vee \bar{x}_2$。在这种情况下，Tarjan 算法不会读取 $\vee \bar{x}_2$。这里的两种情况对应两条数据。

那么我们把这些写下来，就得到了矩阵 Φ'。

$$\Phi' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{5.5}$$

矩阵 5.5 第 8 行的数据对应于未读取 $\vee x_3$ 的情况，而第 9 行的数据对应于未读取 $\vee \bar{x}_2$ 的情况。

显然，在这个矩阵中，$y \neq f$、 f是以 $2SAT$形式存在的环。简单计算可知， $3SAT$ 问题是线性无关的。

实际上，在公式 5.4 的第三个执行子句中，还有另一种方式，即 $\bar{x}_1 \vee \bar{x}_2 \vee x_3 = \bar{x}_1 \vee$ （ $\bar{x}_2 \vee x_3$）。

$\bar{x}_2 \vee x_3$ 被视为 $\bar{x}'_2$，那么 $\bar{x}'_2$ 可被视为具有三个值的逻辑变量，其对应关系如表 1 所示。

TABLE 1. 逻辑值对比

x_2	x_3	$\bar{x}'_2 = \bar{x}_2 \vee x_3$
0	0	0
0	1	1
1	1/0	1

在三值逻辑下， x_2 为真、实际为假和虚拟为假（即，$x_2 = 0$、 x_3 =1）。当 x_2 为真或假时，Tarjan 算法可以正常运行，即能够得到不可满足结论。当 x_2 为假时，由于假节点与真节点之间没有连线，因此不会形成闭环，即环。因此，Tarjan 算法将给出可满足结论。

三值逻辑表明， $2SAT$ 问题，即 SAT 问题的 2 变量、1 个二元运算维度，是一个环结构；而 $3SAT$问题，即 SAT 问题的 3 变量、2 个二元运算维度，则不再可能是环结构。这一事实类似于拓扑学中的现象：二维平面中的圆在三维平面中不构成圆。

对于 kSAT 问题，公式 5.4 中的第三个子句可以进行变换。就像添加 x_3 一样，还可以添加更多 x_k ，也就是说，它可以变换为公式 5.6。

$$f(x_1, x_2, x_3) = x_1 \wedge x_2 \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3 \cdots \vee x_k) \tag{5.6}$$

x_2 公式 5.6 可以视为 k 值逻辑。k 值逻辑与三值逻辑类似，它并不会改变这样的结论：环在不同维度上具有不同的连续性，这与从 $kSAT$ 问题到 $3SAT$ 问题的归约相对应。

6. Conclusions

本文证明了， P在实整数上的本质是一个多项式时间函数；与此同时， NP在复数形式上的本质也是一个多项式时间函数。它们具有不同的维度，从而导致 $P \neq NP$。当然，由于系统、问题或算法有多种表示方式，本文也有多种证明途径可以得出相同的结果。目前，我们至少已知离散表示、系统表示、域数表示、算法表示、拓扑表示和机器表示。这些表示将展示 P和 NP的不同方面。

References

1. Sanjeev Arora 和 Boaz Barak， *Computational complexity: A modern approach*，中国机械工业出版社，2015年。2. Stephen A. Cook， *The complexity of theorem-proving procedures*，STOC，ACM，1971年，第151–158页。3. Oded Goldreich，
Computational complexity: A conceptual perspective，国防工业出版社，2015年。

Abstract. $P?=NP$ 或 $P\ vs\ NP$ 是计算复杂性理论中的核心问题。本文提出了导出矩阵和系统的线性相关性的定义，并讨论了 P和 NP的线性相关性。我们得出结论： P线性相关，并且存在 NP，它线性无关；并以一个属于 NP的3SAT实例为例，即 $P \neq NP$。

South-Central Minzu University, Wuhan, Hubei,430074, China*Current address*: 中国湖北省武汉市洪山区民族大道182号，邮编430074*Email address*: zhengbj@mail.scuec.edu.cn

China Development Bank, Beijing 100080, China*Current address*：北京市西城区太平桥街16号，100080，中国*Email address*： wangweiwu@cdb.cn