Dense Complete Set For NP

Frank Vega
CopSonic, 1471 Route de Saint-Nauphary 82000 Montauban, France

Abstract

A sparse language is a formal language such that the number of strings of length \( n \) is bounded by a polynomial function of \( n \). We create a class with the opposite definition, that is a class of languages that are dense instead of sparse. We define a dense language on \( m \) as a formal language (a set of binary strings) where there exists a positive integer \( n_0 \) such that the counting of the number of strings of length \( n \geq n_0 \) in the language is greater than or equal to \( 2^{n-m} \) where \( m \) is a real number and \( 0 < m \leq 1 \). We call the complexity class of all dense languages on \( m \) as \( DENSE(m) \). We prove that there exists an \( NP–complete \) problem that belongs to \( DENSE(m) \) for every possible value of \( 0 < m \leq 1 \).

1 Summary

In computational complexity theory, a sparse language is a formal language (a set of strings) such that the complexity function, counting the number of strings of length \( n \) in the language, is bounded by a polynomial function of \( n \). The complexity class of all sparse languages is called \( SPARSE \). \( SPARSE \) contains \( TALLY \), the class of unary languages, since these have at most one string of any one length.

Fortune showed in 1979 that if any sparse language is \( coNP–complete \), then \( P = NP \) (this is Fortune’s theorem) [5]. Mahaney used this to show in 1982 that if any sparse language is \( NP–complete \), then \( P = NP \) [6]. A simpler proof of this based on left-sets was given by Ogihara and Watanabe in 1991 [7]. Mahaney’s argument does not actually require the sparse language to be in \( NP \), so there is a sparse \( NP–hard \) set if and only if \( P = NP \) [6].

We create a class with the opposite definition, that is a class of languages that are dense instead of sparse. We show there is a sequence of languages that are in \( NP–complete \), but their density grows as much as we go forward into the iteration of the sequence. The first element of the sequence is a variation of the \( NP–complete \) problem known as \( HAM–CYCLE \) [8]. The next element in the sequence is constructed from this new version of \( HAM–CYCLE \). Indeed, each language is created from its previous one in the sequence. Since the density grows according we move forward into the sequence, then there exists a language so much dense such that its density tends to 0 when the bit-length \( n \) of the binary strings tends to infinity. However, this incredible dense language is still \( NP–complete \).

2 Basic Definitions

Let \( \Sigma \) be a finite alphabet with at least two elements, and let \( \Sigma^* \) be the set of finite strings over \( \Sigma \) [1]. A Turing machine \( M \) has an associated input alphabet \( \Sigma \) [1]. For each string \( w \) in \( \Sigma^* \) there is a computation associated with \( M \) on input \( w \) [1]. We say that \( M \) accepts \( w \) if this computation terminates in the accepting state, that is \( M(w) = “yes” \) [1]. Note that \( M \) fails to accept \( w \) either if this computation ends in the rejecting state, that is \( M(w) = “no” \), or if the computation fails to terminate [1].
The language accepted by a Turing machine $M$, denoted $L(M)$, has an associated alphabet $\Sigma$ and is defined by
\[ L(M) = \{ w \in \Sigma^n : M(w) = \text{"yes"} \}. \]

We denote by $t_M(w)$ the number of steps in the computation of $M$ on input $w$ [1]. For $n \in \mathbb{N}$ we denote by $T_M(n)$ the worst case run time of $M$; that is
\[ T_M(n) = \max \{ t_M(w) : w \in \Sigma^n \} \]
where $\Sigma^n$ is the set of all strings over $\Sigma$ of length $n$ [1]. We say that $M$ runs in polynomial time if there is a constant $k$ such that for all $n$, $T_M(n) \leq n^k + k$ [1]. In other words, this means the language $L(M)$ can be accepted by the Turing machine $M$ in polynomial time. Therefore, $P$ is the complexity class of languages that can be accepted in polynomial time by deterministic Turing machines [4]. A verifier for a language $L$ is a deterministic Turing machine $M$, where
\[ L = \{ w : M(w, c) = \text{"yes" for some string $c$} \}. \]
We measure the time of a verifier only in terms of the length of $w$, so a polynomial time verifier runs in polynomial time in the length of $w$ [1]. A verifier uses additional information, represented by the symbol $c$, to verify that a string $w$ is a member of $L$. This information is called certificate. $NP$ is also the complexity class of languages defined by polynomial time verifiers [8]. If $NP$ is the class of problems that have succinct certificates, then the complexity class $coNP$ must contain those problems that have succinct disqualifications [8]. That is, a “no” instance of a problem in $coNP$ possesses a short proof of its being a “no” instance [8].

A function $f : \Sigma^* \to \Sigma^*$ is a polynomial time computable function if some deterministic Turing machine $M$, on every input $w$, halts in polynomial time with just $f(w)$ on its tape [9]. Let $\{0, 1\}^*$ be the infinite set of binary strings, we say that a language $L_1 \subseteq \{0, 1\}^*$ is polynomial time reducible to a language $L_2 \subseteq \{0, 1\}^*$, written $L_1 \leq_p L_2$, if there is a polynomial time computable function $f : \{0, 1\}^* \to \{0, 1\}^*$ such that for all $x \in \{0, 1\}^*$,
\[ x \in L_1 \text{ if and only if } f(x) \in L_2. \]
An important complexity class is $NP$-complete [4]. A language $L \subseteq \{0, 1\}^*$ is $NP$-complete if
1. $L \in NP$, and
2. $L' \leq_p L$ for every $L' \in NP$.

If $L$ is a language such that $L' \leq_p L$ for some $L' \in NP$-complete, then $L$ is $NP$-hard [4]. Moreover, if $L \in NP$, then $L \in NP$-complete [4]. A principal $NP$-complete problem is $HAM$–$CYCLE$ [4].

A simple graph is an undirected graph without multiple edges or loops [4]. An instance of the language $HAM$–$CYCLE$ is a simple graph $G = (V, E)$ where $V$ is the set of vertices and $E$ is the set of edges, each edge being an unordered pair of vertices [4]. We say $(u, v) \in E$ is an edge in a simple graph $G = (V, E)$ where $u$ and $v$ are vertices. For a simple graph $G = (V, E)$, a simple cycle in $G$ is a sequence of distinct vertices $(v_0, v_1, v_2, ..., v_k)$ such that $(v_k, v_0) \in E$ and $(v_{i-1}, v_i) \in E$ for $i = 1, 2, ..., k$ [4]. A Hamiltonian cycle is a simple cycle of the simple graph which contains all the vertices of the graph. A simple graph that contains a hamiltonian cycle is said to be hamiltonian; otherwise, it is nonhamiltonian [4]. The problem $HAM$–$CYCLE$ asks whether a simple graph is hamiltonian [4].
3 Results

Definition 1. A dense language on $m$ is a formal language (a set of binary strings) where there exists a positive integer $n_0$ such that the counting of the number of strings of length $n \geq n_0$ in the language is greater than or equal to $2^{n-m}$ where $m$ is a real number and $0 < m \leq 1$. The complexity class of all dense languages on $m$ is called DENSE($m$).

Definition 2. A formal language (a set of binary strings) is in DENSE(0) if for every possible value of $0 < m \leq 1$, then the language is always in DENSE($m$).

In this work, we are going to represent the simple graphs with an adjacency-matrix [4]. For the adjacency-matrix representation of a simple graph $G = (V, E)$, we assume that the vertices are numbered $1, 2, \ldots, |V|$ in some arbitrary manner. The adjacency-matrix representation of a simple graph $G$ consists of a $|V| \times |V|$ matrix $A = (a_{i,j})$ such that $a_{i,j} = 1$ when $(i, j) \in E$ and $a_{i,j} = 0$ otherwise [4]. In this way, every simple graph of $k$ vertices could be represented by a binary string of $k^2$ bits.

Observe the symmetry along the main diagonal of the adjacency matrix in this kind of graph that is called simple. We define the transpose of a matrix $A = (a_{i,j})$ to be the matrix $A^T = (a_{j,i})$ given by $a_{i,j}^T = a_{j,i}$. Hence the adjacency matrix $A$ of a simple graph is its own transpose $A = A^T$.

Definition 3. The language NON–SIMPLE contains all the graph that are represented by an adjacency-matrix $A$ such that $A \neq A^T$ or there is some $a_{i,j} = 1$ where $i = j$.

Lemma 4. NON–SIMPLE $\in P$.

Proof. Given a binary string $x$, we can check whether $x$ is an adjacency-matrix which is not equal to its own transpose in time $O(|x|^2)$ just iterating each bit $a_{i,j}$ in $x$ and checking whether $a_{i,j} \neq a_{j,i}$ or $a_{i,j} = 1$ when $i = j$ where $|\ldots|$ represents the bit-length function [4].

Definition 5. The language HAM–CYCLE’ contains all the binary strings $z$ such that $z = xy$, the bit-length of $x$ is equal to $(\sqrt{|z|})^2$ and $x \in$ HAM–CYCLE or $x \in$ NON–SIMPLE where $y$ could be the empty string when $\ldots$ and $[\ldots]$ represent the bit-length function and the floor function respectively.

Lemma 6. HAM–CYCLE’ $\in NP$–complete.

Proof. Given a binary string $z$ such that $z = xy$ and the bit-length of $x$ is equal to $(\sqrt{|z|})^2$, we can decide in polynomial time whether $x \notin$ NON–SIMPLE just verifying when $x = x^T$ and $a_{i,i} = 0$ for all vertex $i$. In this way, we can reduce in polynomial time a simple graph $G = (V, E)$ of $k$ vertices encoded as the binary string $x$ such that when $x$ has $k^2$ bits and $x \notin$ NON–SIMPLE then

$$x \in \text{HAM–CYCLE if and only if } xy \in \text{HAM–CYCLE’}$$

where $y$ could be the empty string. In this way, we can reduce in polynomial time each element of HAM–CYCLE to some element of HAM–CYCLE’. Therefore, HAM–CYCLE’ is in NP–hard. Moreover, we can check in polynomial time over a binary string $z$ such that $z = xy$ and the bit-length of $x$ is equal to $(\sqrt{|z|})^2$ whether $x \in$ HAM–CYCLE or $x \in$ NON–SIMPLE since HAM–CYCLE $\in NP$ and NON–SIMPLE $\in NP$ because of $P \subseteq NP$ [8]. Consequently, HAM–CYCLE’ is in NP. Hence, HAM–CYCLE’ $\in NP$–complete.
Lemma 7. \( HAM\text{-CYCLE}' \in DENSE(1) \). This would mean the existence of a sufficiently large positive integer \( n'_0 \) such that all the binary strings of length \( n \geq n'_0 \) which belong to \( HAM\text{-CYCLE}' \) are more than or equal to \( 2^{n-1} \) elements.

Proof. OEIS A000088 gives some number of graphs on \( n \) unlabeled points [10]. For 8 points there are 12346 so just over half the graphs on 8 points are Hamiltonian [10]. For 12 points, there are 152522187830 Hamiltonian graphs out of 165091172592 which would claim that over 92% of the 12 point graphs are Hamiltonian [10]. For \( n = 2 \) there are two graphs, neither of which is Hamiltonian [10]. For \( n < 8 \) over half the graphs are not Hamiltonian [10]. It does not seem surprising that once \( n \) gets large most graphs are Hamiltonian [10].

Choosing a graph on \( n \) vertices at random is the same as including each edge in the graph with probability \( \frac{1}{2} \), independently of the other edges [2]. You get a more general model of random graphs if you choose each edge with probability \( p \) [2]. This model is known as \( G_{n,p} \) [2]. It turns out that for any constant \( p > 0 \), the probability that \( G_{n,p} \) contains a Hamiltonian cycle tends to 1 when \( n \) tends to infinity [2]. In fact, this is true whenever \( p > \frac{c \log n}{n} \) for some constant \( c \). In particular this is true for \( p = \frac{1}{2} \), which is our case [2].

For all the binary strings \( z \) such that \( z = xy \) and the bit-length of \( x \) is equal to \( (\lceil \sqrt{|z|} \rceil)^2 \), the amount of elements of size \( |z| \) in \( HAM\text{-CYCLE}' \) is equal to the number of binary strings \( x \in HAM\text{-CYCLE} \) or \( x \in NON\text{-SIMPLE} \) of size \( (\lceil \sqrt{|z|} \rceil)^2 \) multiplied by \( 2^{|z|-(\lceil \sqrt{|z|} \rceil)^2} \). Since the number of Hamiltonian graphs increases as much as we go further on \( n \), it does not seem surprising either that once \( n \) gets large most binary strings belong to \( HAM\text{-CYCLE}' \). Moreover, the amount of binary strings which have some bit-length \( k^2 \) and belongs to \( NON\text{-SIMPLE} \) is considerably superior to the amount of strings with the same bit-length which are valid simple graphs. Actually, we can affirm for a sufficiently large positive integer \( n'_0 \), all the binary strings of length \( n \geq n'_0 \) which belong to \( HAM\text{-CYCLE}' \) are indeed more than or equal to \( 2^{n-1} \) elements. In this way, we show that \( HAM\text{-CYCLE}' \in DENSE(1) \). ▶

Definition 8. We will define a sequence of languages \( HAM\text{-CYCLE}'_k \) for every possible integer \( 1 \leq k \). We state \( HAM\text{-CYCLE}'_1 \) as the language \( HAM\text{-CYCLE}' \). Recursively, from a language \( HAM\text{-CYCLE}'_k \), we define \( HAM\text{-CYCLE}'_{k+1} \) as follows: A binary string \( xy \) complies with \( xy \in HAM\text{-CYCLE}'_{k+1} \) if and only if \( x \) and \( y \) are binary strings, \( x \in HAM\text{-CYCLE}'_k \) or \( y \in HAM\text{-CYCLE}'_k \) such that \( |x| = \lfloor \frac{|xy|}{2} \rfloor \) where \( \lfloor \ldots \rfloor \) represents the bit-length function and \( \lfloor \ldots \rfloor \) is the floor function.

Lemma 9. For every integer \( 1 \leq k \), \( HAM\text{-CYCLE}'_k \in NP \).

Proof. This is true for \( k = 1 \) as we see in Lemma 6. Every string \( xy \) which belongs to \( HAM\text{-CYCLE}'_2 \) complies with \( x \in HAM\text{-CYCLE}'_1 \) or \( y \in HAM\text{-CYCLE}'_1 \) such that \( |x| = \lfloor \frac{|xy|}{2} \rfloor \). Moreover, every string \( xy \) which belongs to the language \( HAM\text{-CYCLE}'_3 \) complies with \( x \in HAM\text{-CYCLE}'_2 \) or \( y \in HAM\text{-CYCLE}'_2 \) such that \( |x| = \lfloor \frac{|xy|}{2} \rfloor \). Furthermore, we can extend this property for every positive integer \( k > 3 \) in \( HAM\text{-CYCLE}'_k \). Indeed, \( HAM\text{-CYCLE}'_k \) is in \( NP \) for every integer \( 1 \leq k \), since the verification of whether the two substrings are indeed elements of \( HAM\text{-CYCLE}'_{k-1} \) can be done in polynomial time with the appropriated certificates using the induction on \( k \). ▶

Theorem 10. For every integer \( 1 \leq k \), \( HAM\text{-CYCLE}'_k \in NP\text{-complete} \).

Proof. This is true for \( k = 1 \) by the Lemma 6. Let’s assume it is valid for some positive integer \( 1 \leq k' \). Let’s prove this for \( k' + 1 \). We already know the adjacency-matrix of \( n^2 \) zeros represents a simple graph of \( n \) vertices which does not contain any edge. This kind of a simple graph does not belong to \( HAM\text{-CYCLE}'_1 \). As a consequence, this string will
not belong to any $\text{HAM–CYCLE}'_k$, because its substrings of a quadratic length are also adjacency-matrix of only zeros. Suppose, we have an instance $y$ of $\text{HAM–CYCLE}'_k$. We can reduce $y$ in $\text{HAM–CYCLE}'_k$, to $zy$ in $\text{HAM–CYCLE}'_{k+1}$ such that

$$y \in \text{HAM–CYCLE}'_k \text{ if and only if } zy \in \text{HAM–CYCLE}'_{k+1}$$

where the binary string $z$ is exactly a sequence of $\lfloor \frac{|zy|}{2} \rfloor$ zeros. We can do this since we already know $z \notin \text{HAM–CYCLE}'_k$. Certainly, if the membership $zy \in \text{HAM–CYCLE}'_{k+1}$ is true, $z \notin \text{HAM–CYCLE}'_k$ and $|z| = \lfloor \frac{|zy|}{2} \rfloor$, then $y \in \text{HAM–CYCLE}'_k$ also holds according to the Definition 8. Since this reduction remains in polynomial time for every positive integer $1 \leq k'$, then we show that $\text{HAM–CYCLE}'_{k'+1}$ is in $\text{NP}$–hard. Moreover, $\text{HAM–CYCLE}'_{k'+1}$ is also in $\text{NP}$–complete, because of the Lemma 9.

**Theorem 11.** For every integer $1 \leq k$, if the language $\text{HAM–CYCLE}'_k$ is in $\text{DENSE}(k')$ for every instance of bit-length $n' \geq n_0$, then $\text{HAM–CYCLE}'_{k+1}$ is in $\text{DENSE}(\frac{k'}{2})$ for every instance of bit-length $n' \geq 2 \times n_0$.

**Proof.** If the language $\text{HAM–CYCLE}'_k$ is in $\text{DENSE}(k')$ for every instance of bit-length $n' \geq n_0$, then for every integer $n \geq n_0$ the amount of elements of size $n+i$ in $\text{HAM–CYCLE}'_{k+1}$ (where $i \geq n_0$ and $i = \lfloor \frac{n+i}{2} \rfloor$) is greater than or equal to

$$2^{i-k'} \times 2^n + 2^{n-k'} \times (2^i - 2^{i-k'})$$

This is because there must be more than or equal to $2^{i-k'}$ elements of size $i$ in $\text{HAM–CYCLE}'_k$ which are prefixes of the binary strings of size $n+i$ in the language $\text{HAM–CYCLE}'_{k+1}$. We multiply that amount by $2^n$ since this is the number of different combinations of suffixes with length $n$ in the binary strings of size $n+i$. Moreover, there must be more than or equal to $2^{n-k'}$ elements of size $n+i$ in $\text{HAM–CYCLE}'_k$ which are suffixes of the binary strings of size $n+i$ in $\text{HAM–CYCLE}'_{k+1}$. We multiply that amount by $(2^i - 2^{i-k'})$ since this is the number of different combinations of prefixes with length $i$ in the binary strings of size $n+i$ just avoiding to count the previous prefixes twice. If we join both properties, we obtain the sum described by the formula above.

Indeed, this formula can be simplified to

$$2^{n+i-k'} + 2^{n+i-k'} \times (2^0 - 2^{-k'})$$

and extracting a common factor we obtain

$$2^{n+i-k'} \times (1 + (1 - 2^{-k'}))$$

which is equal to

$$2^{n+i-k'} \times (2 - \frac{1}{2^k}).$$

Nevertheless, for every real number $0 < k' \leq 1$ we have that

$$(2 - \frac{1}{2^k}) \geq 2^{k'}.$$

Certainly, if we multiply both member of the inequality by $2^{k'}$, we obtain

$$(2^{k'+1} - 1) \geq 2^{k' + \frac{k'}{2}}.$$
which is equivalent to

\[ 2^{k'} \times (2 - 2^{k'}) \geq 1 \]

that it is true for every real number \( 0 < k' \leq 1 \). We can check in the Figure 1 that the function \( f(x) = 2^x \times (2 - 2^{k'}) \) is greater than or equal to 1 over the interval \([0, 1]\). Thus

\[ 2^{n+i-k'} \times (2 - \frac{1}{2^{k'}}) \geq 2^{n+i-k'} \times 2^{k'} \]

where

\[ 2^{n+i-k'} \times 2^{k'} = 2^{n+i-(k'-\frac{k'}{2})} = 2^{n+i-k'} \times 2^{k'} \]

Since there are more than or equal to \( 2^n-(\frac{k'}{2}) \) elements of the language \( HAM-CYCLE'_{k+1} \) with length \( n' \geq 2 \times n_0 \) therefore, we show that \( HAM-CYCLE'_{k+1} \) is in \( DENSE(\frac{k'}{2}) \) for every instance of bit-length \( n' \geq 2 \times n_0 \).

**Lemma 12.** \( HAM-CYCLE'_{k} \in DENSE(\frac{1}{2^k}) \) for every instance of bit-length \( n \geq 2^{k-1} \times n_0' \), where the constant \( n_0' \) is the positive integer used in the Definition 1 and Lemma 7 for \( HAM-CYCLE' \).

**Proof.** According to the Lemma 7, \( HAM-CYCLE'_{1} \) is in \( DENSE(1) \) for every instance of bit-length \( n \geq 2^0 \times n_0' = n_0' \). Consequently, due to Theorem 11, \( HAM-CYCLE'_{2} \) is in \( DENSE(\frac{1}{2}) \) for every instance of bit-length \( n \geq 2^1 \times n_0' \). Moreover, \( HAM-CYCLE'_{3} \) is in \( DENSE(\frac{1}{4}) \) for every instance of bit-length \( n \geq 2^2 \times n_0' \) and so forth . . . and thus, for every language \( HAM-CYCLE'_{k} \), we have that \( HAM-CYCLE'_{k} \in DENSE(\frac{1}{2^k}) \) for every instance of bit-length \( n \geq 2^{k-1} \times n_0' \).

**Definition 13.** We will define a language \( HAM-CYCLE'_{\infty} \) as follows: A binary string \( x \) complies with \( x \in HAM-CYCLE'_{\infty} \) if and only if we obtain that \( x \in HAM-CYCLE'_{k} \) and \( 2^{k-1} \times n_0' \leq |x| < 2^k \times n_0' \) where \( |. . .| \) represents the bit-length function and the constant \( n_0' \) is the positive integer used in the Definition 1 and Lemma 7 for \( HAM-CYCLE' \).

**Lemma 14.** \( HAM-CYCLE'_{\infty} \in NP \).

**Proof.** We can calculate the value of \( k \) from some binary string \( x \) that is approximately \( \lceil \log_2(\frac{|x|}{n_0'}) \rceil \), where \( \lceil . . . \rceil \) is the ceiling function. In this way, we should know if
When a language is sparse, then its complement is in $\text{DENSE}(0)$ [6]. Indeed, the sparse languages are called sparse because there are a total of $2^n$ strings of length $n$, and if a language only contains polynomially many of these, then the proportion of strings of length $n$ that it contains rapidly goes to zero as $n$ grows (which means its complement should be in $\text{DENSE}(0)$) [6]. In addition, according to Theorem 15, the complement of this language $\text{HAM–CYCLE}'_{\infty}$ must be in $\text{coNP–complete}$, because of the complement of the $\text{NP–complete}$ problems are complete for $\text{coNP}$ [8]. In 1999, Jin-Yi Cai and D. Sivakumar, building on work by Ogihara, showed that if there exists a sparse $P$-complete problem, then $\text{LOGSPACE} = P$ [3]. We might extend the proof of this paper to show the same result on $P$. Certainly, we might only need to find some $P$-complete which belongs to $\text{DENSE}(1)$ because the $P$-completeness is closed under complement [8]. Indeed, the other steps of that possible proof might be similar to the arguments that we follow in this paper.
References


