Lezione di Informatica Teorica: Classi di Complessità Co-NP, EXP, NEXP

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Introduzione alle Classi di Complessità

Riprendiamo il concetto di classe **NP**. Ricordiamo che un problema di decisione L appartiene alla classe NP se esiste una Macchina di Turing non deterministica (NTM) che decide L in tempo polinomiale. Intuitivamente, le istanze "sì" di un problema in NP ammettono un "certificato" conciso (polinomiale) che può essere verificato in tempo polinomiale da una Macchina di Turing deterministica.

1.1 Problema UNSAT

Consideriamo il problema **UNSAT**: Data una formula booleana f in forma normale congiuntiva (CNF), stabilire se f *non è soddisfacibile*.

Definizione 1 (Linguaggio UNSAT)**.** *Il linguaggio UNSAT è l'insieme delle stringhe che codificano formule booleane in CNF che non sono soddisfacibili.*

$$UNSAT = \{\langle f \rangle \mid f \text{ è una formula in CNF non soddisfacibile}\}$$

Una formula non soddisfacibile è una formula per la quale non esiste alcuna assegnazione di verità alle variabili che la renda vera.

Domanda: UNSAT appartiene alla classe NP? **Risposta intuitiva:** No. Per rispondere "sì" (la formula non è soddisfacibile), dovremmo essere sicuri che *nessuna* possibile assegnazione di verità la soddisfa. Non possiamo semplicemente "indovinare" un'assegnazione e verificare che non la soddisfa, perché un'assegnazione che non soddisfa la formula non prova che la formula sia non soddisfacibile; ne servirebbe una che la soddisfa per dire "no".

Se una macchina non deterministica "indovinasse" un'assegnazione di verità per le variabili di f, verificasse che tale assegnazione *non* soddisfa f, e poi rispondesse "sì", tale macchina deciderebbe in realtà il complemento del linguaggio delle tautologie (formule sempre vere).

2 La Classe Co-NP

Introduciamo una nuova classe di complessità, **Co-NP**, che formalizza il tipo di problemi come UNSAT.

Definizione 2 (Classe Co-NP)**.** *La classe Co-NP è l'insieme dei linguaggi L tali che il loro complemento $\overline{L}$ appartiene alla classe NP.*

$$Co\text{-}NP = \{L \mid \overline{L} \in NP\}$$

Esempio 1: UNSAT UNSAT appartiene a Co-NP, poiché il suo complemento $\overline{\text{UNSAT}}$ è il problema SAT (Satisfiability), e SAT è un problema NP-completo, quindi appartiene a NP.

Importante: Co-NP *non* è il complemento di NP. Il complemento di NP conterrebbe tutti i problemi fuori da NP, inclusi problemi indecidibili, problemi non ricorsivamente enumerabili, ecc. Co-NP è invece una classe ben definita di problemi decidibili.

2.1 Intuizione per Co-NP

In modo speculare all'intuizione per NP:

- Un linguaggio L è in **NP** se le sue istanze "sì" possono essere verificate efficientemente tramite un certificato conciso.
- Un linguaggio L è in **Co-NP** se le sue istanze "no" possono essere verificate efficientemente tramite un certificato conciso.

Per i problemi in Co-NP, siamo in grado di rispondere "no" in modo efficiente, fornendo un "certificato di rifiuto".

Esempio 1 (UNSAT - Intuizione Co-NP)**.** *Per decidere UNSAT, se la formula f non è non soddisfacibile (cioè è soddisfacibile), possiamo fornire un certificato: un'assegnazione di verità che soddisfa f. Questo certificato permette di rispondere "no" in tempo polinomiale.*

Esempio 2 (TAUTOLOGY)**.** *Il problema TAUTOLOGY è decidere se una data formula booleana in CNF è una tautologia (cioè sempre vera per ogni assegnazione di verità). TAUTOLOGY appartiene a Co-NP. Per rispondere "no" a una formula che non è una tautologia, basta fornire un'assegnazione di verità che la renda falsa. Questa verifica è polinomiale.*

3 Relazioni tra P, NP e Co-NP

Mentre per le classi R e Co-R sappiamo che sono distinte e che la loro intersezione è R ($R \cap$ Co-R $= R$), per P, NP e Co-NP la situazione è molto meno chiara. La maggior parte delle relazioni sono problemi aperti.

- Non sappiamo se NP $=$ Co-NP o se NP $\neq$ Co-NP. Questa è una questione aperta. L'ipotesi corrente è che siano distinte.
- Non sappiamo se P $=$ NP $\cap$ Co-NP. Sappiamo che P $\subseteq$ NP $\cap$ Co-NP.

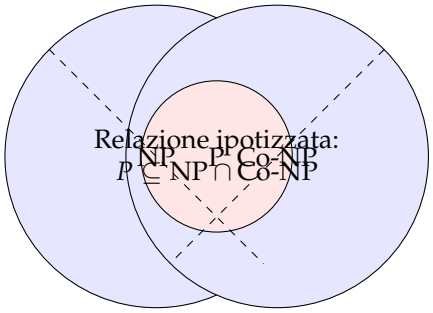

Figura 1: Relazioni ipotizzate tra P, NP e Co-NP.

Teorema 1 (Relazione NP e Co-NP)**.** *NP $=$ Co-NP se e solo se esiste un linguaggio L NP-completo tale che $L \in$ Co-NP.*

Dimostrazione. **Parte 1: Se NP $=$ Co-NP allora esiste L NP-completo tale che $L \in$ Co-NP.** Se NP $=$ Co-NP, allora qualsiasi linguaggio L NP-completo (che per definizione appartiene a NP) apparterrà anche a Co-NP. Questa direzione è banale.

Parte 2: Se esiste L NP-completo tale che $L \in$ Co-NP allora NP $=$ Co-NP. Per dimostrare che NP $=$ Co-NP, dobbiamo mostrare due inclusioni: NP $\subseteq$ Co-NP e Co-NP $\subseteq$ NP.

1. **Dimostriamo NP $\subseteq$ Co-NP:** Sia L' un linguaggio qualsiasi in NP. Il nostro obiettivo è mostrare che $L' \in$ Co-NP, il che significa $\overline{L'} \in$ NP.

   - Poiché $L' \in$ NP e L è NP-completo per ipotesi, sappiamo che L' si riduce polinomialmente a L (cioè $L' \leq_P L$). Questo significa che esiste una funzione di trasformazione f calcolabile in tempo polinomiale tale che per ogni stringa w: $w \in L' \iff f(w) \in L$
   - Questa equivalenza può essere riscritta in termini di complementi: $w \notin L' \iff f(w) \notin L$ Ciò implica: $w \in \overline{L'} \iff f(w) \in \overline{L}$ Questa relazione mostra che $\overline{L'}$ si riduce polinomialmente a $\overline{L}$ (cioè $\overline{L'} \leq_P \overline{L}$).
   - Per ipotesi, $L \in$ Co-NP. Per definizione di Co-NP, questo significa che $\overline{L} \in$ NP.
   - Poiché $\overline{L'}$ si riduce a $\overline{L}$ (che è in NP), e la classe NP è chiusa rispetto a riduzioni polinomiali (se $A \leq_P B$ e $B \in$ NP allora $A \in$ NP), allora $\overline{L'} \in$ NP.
   - Dato che $\overline{L'} \in$ NP, per definizione di Co-NP, $L' \in$ Co-NP.

   Abbiamo quindi dimostrato che qualsiasi $L' \in$ NP è anche in Co-NP, da cui NP $\subseteq$ Co-NP.

2. **Dimostriamo Co-NP $\subseteq$ NP:** Sia L' un linguaggio qualsiasi in Co-NP. Il nostro obiettivo è mostrare che $L' \in$ NP.

   - Poiché $L' \in$ Co-NP, per definizione, $\overline{L'} \in$ NP.
   - Per ipotesi, L è un linguaggio NP-completo. Poiché $\overline{L'} \in$ NP e L è NP-completo, sappiamo che $\overline{L'}$ si riduce polinomialmente a L (cioè $\overline{L'} \leq_P L$).
   - Similmente al punto precedente, questa riduzione implica che L' si riduce polinomialmente a $\overline{L}$ (cioè $L' \leq_P \overline{L}$).
   - Per ipotesi, $L \in$ Co-NP, il che significa $\overline{L} \in$ NP.
   - Poiché L' si riduce a $\overline{L}$ (che è in NP), e NP è chiusa rispetto a riduzioni polinomiali, allora $L' \in$ NP.

   Abbiamo quindi dimostrato che qualsiasi $L' \in$ Co-NP è anche in NP, da cui Co-NP $\subseteq$ NP.

Dato che NP $\subseteq$ Co-NP e Co-NP $\subseteq$ NP, concludiamo che NP $=$ Co-NP. □

3.1 Problemi Co-NP-hard

Analogamente a NP-hard, definiamo Co-NP-hard.

Definizione 3 (Co-NP-hard)**.** *Un problema L è **Co-NP-hard** se $\overline{L}$ è NP-hard.*

I problemi Co-NP-hard sono almeno difficili quanto tutti i problemi in Co-NP. UNSAT è un esempio di problema Co-NP-completo.

4 Il Problema della Fattorizzazione (FACTOR)

Il problema della fattorizzazione di numeri interi è un problema centrale in crittografia. Sebbene il problema di "produrre la fattorizzazione" non sia un problema di decisione, possiamo definirne una versione decisionale.

Definizione 4 (Linguaggio FACTOR)**.** *Il linguaggio FACTOR è l'insieme delle coppie $\langle n, k\rangle$ tali che n è un intero naturale e n ha almeno un fattore primo p tale che $p \leq k$.*

$$FACTOR = \{\langle n, k\rangle \mid n \in \mathbb{N}, \exists p \in \mathbb{P} \text{ t.c. } p \leq k \text{ e } p \text{ divide } n\}$$

Esempio 3.
- $\langle 175, 6\rangle \in$ *FACTOR perché* $175 = 5 \cdot 5 \cdot 7$*. Il fattore primo* $5 \leq 6$*, e* 5 *divide* 175*.*
- $\langle 175, 4\rangle \notin$ *FACTOR perché nessuno dei fattori primi di* 175 $(5, 7)$ *è minore o uguale a* 4*.*

Teorema 2. *FACTOR* $\in$ *NP* $\cap$ *Co-NP.*

Dimostrazione. **Parte 1: FACTOR $\in$ NP** Per dimostrare che FACTOR è in NP, dobbiamo mostrare che esiste una NTM che lo decide in tempo polinomiale.

- **Guess:** La NTM "indovina" un numero primo candidato p. Dato che $p \leq k$ (e $k \leq n$), p non può essere "troppo grande" rispetto alla dimensione dell'input $\langle n, k\rangle$, quindi il guess ha una dimensione polinomiale rispetto all'input.
- **Verifica (Check):**
  1. Verificare che $p \leq k$. (Tempo polinomiale).
  2. Verificare che p sia un numero primo. L'algoritmo AKS (Agrawal-Kayal-Saxena, 2003) testa la primalità in tempo polinomiale deterministico.
  3. Verificare che p divida n (cioè $n \pmod p = 0$). La divisione tra interi può essere eseguita in tempo polinomiale rispetto alla dimensione (numero di bit) degli operandi.

  Tutti i passaggi di verifica sono polinomiali, quindi FACTOR $\in$ NP.

Parte 2: FACTOR $\in$ Co-NP Per dimostrare che FACTOR è in Co-NP, dobbiamo mostrare che il suo complemento $\overline{\text{FACTOR}}$ è in NP.

Definizione 5 (Linguaggio $\overline{\text{FACTOR}}$)**.** *$\overline{FACTOR}$ è l'insieme delle coppie $\langle n, k\rangle$ tali che n è un intero naturale e tutti i fattori primi p di n sono maggiori di k.*

$$\overline{FACTOR} = \{\langle n, k\rangle \mid n \in \mathbb{N}, \forall p \in \mathbb{P} \text{ t.c. } p \text{ divide } n \implies p > k\}$$

Per dimostrare che $\overline{\text{FACTOR}} \in$ NP:

- **Guess:** La NTM "indovina" l'intera fattorizzazione prima di n: $p_1, p_2, \ldots, p_m$. Il numero di fattori m è al più logaritmico rispetto a n (poiché $2^m \leq n \implies m \leq \log_2 n$). Quindi la dimensione del guess è polinomiale nella dimensione dell'input.
- **Verifica (Check):**
  1. Verificare che $p_i > k$ per ogni $i = 1, \ldots, m$. (Tempo polinomiale).
  2. Verificare che ogni p_i sia un numero primo (con l'algoritmo AKS). Questo richiede m test di primalità, per un costo totale polinomiale.
  3. Verificare che il prodotto $p_1 \cdot p_2 \cdot \cdots \cdot p_m$ sia uguale a n. La moltiplicazione di m numeri può essere eseguita in tempo polinomiale.

  Tutti i passaggi di verifica sono polinomiali, quindi $\overline{\text{FACTOR}} \in$ NP.

Poiché $\overline{\text{FACTOR}} \in$ NP, per definizione, FACTOR $\in$ Co-NP.
Combinando le due parti, FACTOR $\in$ NP $\cap$ Co-NP. □

4.1 Implicazioni del posizionamento di FACTOR

Il problema della fattorizzazione è fondamentale per la crittografia (e.g., RSA si basa sulla sua presunta difficoltà).

- FACTOR $\notin$ P (ipotesi): Nessuno è ancora riuscito a trovare un algoritmo polinomiale deterministico per la fattorizzazione. Se lo fosse, molti schemi crittografici sarebbero compromessi.
- FACTOR $\notin$ NP-completo (ipotesi): Poiché FACTOR $\in$ NP $\cap$ Co-NP, se fosse NP-completo, implicherebbe NP $=$ Co-NP (per il teorema precedente). Dato che si ipotizza NP $\neq$ Co-NP, si ipotizza anche che FACTOR non sia NP-completo.

Questo colloca FACTOR in una "terra di mezzo": un problema che si presume non sia in P (non semplice) ma nemmeno NP-completo (non il più difficile in NP).

Nota sui computer quantistici: L'algoritmo di Shor per macchine quantistiche risolve FACTOR in tempo polinomiale. Questo non implica che P $=$ NP o che i problemi NP-completi siano risolvibili in tempo polinomiale da macchine quantistiche. L'algoritmo di Shor dimostra che FACTOR non è "intrinsecamente" difficile per tutte le classi di calcolo (come quella quantistica), ma non confuta le congetture sulla difficoltà dei problemi NP-completi per i computer classici.

5 Classi di Complessità Superiori: EXP e NEXP

Oltre alle classi P, NP, e Co-NP, esistono classi di complessità che considerano tempi di esecuzione maggiori, in particolare esponenziali.

5.1 Classe EXP (Exponential Time)

Definizione 6 (Classe EXP)**.** *La classe **EXP** (o **EXPTIME**) è l'insieme dei linguaggi che possono essere decisi da una Macchina di Turing deterministica in tempo esponenziale, ovvero $O(2^{n^c})$ per qualche costante $c \geq 1$.*

$$EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$$

Relazioni:

- P $\subseteq$ EXP: Ovvio, poiché un tempo polinomiale è anche un tempo esponenziale.
- P $\neq$ EXP: Questo è un risultato dimostrato dal Teorema della Gerarchia Temporale (Time Hierarchy Theorem). Ci sono problemi risolvibili in tempo esponenziale che non sono risolvibili in tempo polinomiale.
- NP $\subseteq$ EXP: Una Macchina di Turing non deterministica che opera in tempo polinomiale può essere simulata da una Macchina di Turing deterministica in tempo esponenziale (esplorando l'albero di computazione).
- Co-NP $\subseteq$ EXP: Se $L \in$ Co-NP, allora $\overline{L} \in$ NP. Poiché NP $\subseteq$ EXP, allora $\overline{L} \in$ EXP. Le classi deterministiche come EXP sono chiuse sotto complemento (se un problema è in EXP, anche il suo complemento lo è), quindi $L \in$ EXP.
- NP vs EXP: È un problema aperto se NP $=$ EXP o NP $\neq$ EXP. Si ipotizza che siano distinte.

5.2 Classe NEXP (Non-deterministic Exponential Time)

Definizione 7 (Classe NEXP)**.** *La classe **NEXP** (o **NEXPTIME**) è l'insieme dei linguaggi che possono essere decisi da una Macchina di Turing non deterministica in tempo esponenziale, ovvero $O(2^{n^c})$ per qualche costante $c \geq 1$.*

$$NEXP = \bigcup_{c \geq 1} NTIME(2^{n^c})$$

Relazioni:

- EXP $\subseteq$ NEXP: Una DTM è un caso speciale di NTM.
- EXP vs NEXP: È un problema aperto se EXP $=$ NEXP o EXP $\neq$ NEXP.
- NP $\subsetneq$ NEXP: Il Teorema della Gerarchia Temporale implica che le classi temporali non deterministiche con differenze esponenziali sono distinte.

5.3 Caratterizzazione di NEXP basata sui Certificati

Similmente a NP, NEXP può essere caratterizzata in termini di certificati.

Definizione 8 (Caratterizzazione Certificata di NEXP)**.** *Un linguaggio L appartiene a **NEXP** se le sue istanze "sì" sono caratterizzate da certificati di taglia esponenziale (rispetto alla dimensione dell'input) che possono essere verificati da una Macchina di Turing deterministica in tempo polinomiale nella taglia combinata dell'input e del certificato.*

Formalmente, per un linguaggio L, se $w \in L$, esiste un certificato C tale che:

- $|C| = O(2^{|w|^k})$ per qualche $k \geq 1$.
- Esiste una DTM che accetta $\langle w, C \rangle$ in tempo $P(|w| + |C|)$ per qualche polinomio P.

Questa è la ragione per cui la verifica è polinomiale nella taglia combinata: se fosse esponenziale nella taglia del certificato, il costo totale sarebbe doppiamente esponenziale rispetto all'input.

6 Riepilogo delle Relazioni tra Classi di Complessità

Le relazioni tra le classi di complessità sono in gran parte ancora problemi aperti, ma ci sono alcune inclusioni e separazioni note o ipotizzate.

- P $\subseteq$ NP $\subseteq$ EXP $\subseteq$ NEXP.
- P $\subseteq$ NP $\cap$ Co-NP.
- P $\neq$ EXP (confermato dal Teorema della Gerarchia Temporale).
- NP $\neq$ NEXP (confermato dal Teorema della Gerarchia Temporale).
- Le relazioni P vs NP, NP vs Co-NP, NP vs EXP, EXP vs NEXP sono tutte problemi aperti. Le congetture più comuni sono che siano tutte inclusioni strette ($\neq$).

Le classi di complessità temporale deterministiche (come P, EXP) sono chiuse sotto complemento, mentre quelle non deterministiche (come NP, NEXP) non si sa con certezza.

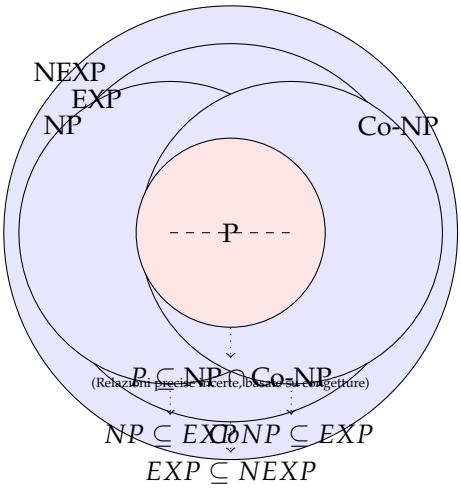

Figura 2: Panoramica delle relazioni tra le classi di complessità (ipotesi correnti).